A Reconfigurable Hardware Testbed for Elastically-Coupled Systems

by

Nicholas Scott Hardman

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in Electrical Engineering

University of Washington

1997

Approved by: ______________________
(Chairperson of Supervisory Committee)

Deirdre R. Meldrum
____

____

Program Authorized
to Offer Degree: Electrical Engineering

Date: ______________________

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
4 August 1998

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
A RECONFIGURABLE HARDWARE TESTBED FOR ELASTICALLY-COUPLED SYSTEMS

5. FUNDING NUMBERS

6. AUTHOR(S)
Nicholas Scott Hardman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Washington

8. PERFORMING ORGANIZATION REPORT NUMBER
98-040

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With AFI 35-205/AFIT Sup 1

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS

15. NUMBER OF PAGES
108

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT

18. SECURITY CLASSIFICATION OF THIS PAGE

19. SECURITY CLASSIFICATION OF ABSTRACT

20. LIMITATION OF ABSTRACT

TABLE OF CONTENTS

TABLE OF CONTENTS i
LIST OF FIGURES iv
LIST OF TABLES vi
Chapter 1 INTRODUCTION 1
1.1 Motivation 1
1.2 Purpose 2
1.3 Previous Work 5
1.4 Thesis Outline 5
Chapter 2 TESTBED DESCRIPTION 7
2.1 Testbed Layout 7
2.2 Xmath Software 8
2.3 AC100 system 9
2.4 Servo Amplifier 11
2.5 Power Supply 14
2.6 Brushless DC Servo Motor 15
2.7 Gearing 20
2.8 Sensors 22
2.9 Reaction Mass Actuator 23
2.10 Ball Bearing Sliders 25
2.11 Damper 26
2.12 Springs 26
2.12.1 Mass Displacement 27
2.12.2 Resonant Frequency 28
2.13 Masses 28
2.14 Control Box 28
2.15 Machined Components 30
Chapter 3 MODEL DEVELOPMENT 32

3.1 Modeling the RECS Testbed 32
3.2 Open Loop Configuration 32
3.2.1 Physical System 33
3.2.2 Mathematical Model Derivation 33
3.2.3 State Space Formulation 34
3.2.3.1 Definitions 34
3.2.3.2 State Matrices 35
3.2.4 Model with Friction Compensation 37
3.2.4.1 Equations 37
3.2.4.2 State Matrices 38
3.3 Closed Loop Configuration 38
3.3.1 Physical System 39
3.3.2 Mathematical Model Derivation 40
3.3.3 State Space Formulation 40
3.3.3.1 Definitions 40
3.3.3.2 State Matrices 41
3.3.4 Model with Friction Compensation 42
3.3.5 State Matrices with Friction Compensation 43
3.4 Nonlinearities 44
Chapter 4 SIMULATIONS AND MODEL COMPARISON 45
4.1 Matlab Simulations 45
4.2 Model Comparison 49
Chapter 5 CONTROLLER DEVELOPMENT 53
5.1 Configuration 1: System Identification 54
5.2 Configuration 1: Closed Loop Operation 53
5.3 Configuration 2: Closed Loop Operation 55
Chapter 6 CONCLUSIONS AND FUTURE DISCUSSION 56
6.1 Conclusions 56

6.2 Future Development 56

6.2.1 GUI Interactive Program 56

6.2.2 Future Study 57

Appendix A WIRING 59

Appendix B AC100/Matrix$_x$ IMPLIMENTATION 69

B.1 RECS Testbed Checklist 69

B.2 AC100 Layout 69

Appendix C Matlab Simulation Program Files 82

C.1 Model Simulation File Index 82

C.2 File: Master.m 83

C.3 File: bode_dsa.m 85

C.4 File: bodeplus.m 85

C.5 File: config1.m 86

C.6 File: config2.m 90

C.7 File: c1sim.m 93

C.8 File: c2sim.m 96

C.9 File: RECSgui.m 99

Appendix D Machined Parts Blueprints 106

LIST OF FIGURES

1.1: Full Configurations for Both Forms 3
1.2: UW CRSL RECS Testbed 4
2.1: Block Diagram of RECS Testbed 7
2.2: Xmath GUI in RealSim 8
2.3: Computer Setup.. 11
2.4: Servo Amplifier Schematic................................ 12
3.5: Servo Amplifier Card 13
3.6: Power Supply in NEMA 12 Enclosure with AMP Connector 14
2.7: DC Motor Drawing 16
2.8: Motor Envelope 16
2.9: Motor model 18
2.10: Top and Side View of Slider #1 21
2.11: HEDS-9200 Optical Encoder 23
2.12: Reaction Mass Actuator 24
2.13: Spring Analysis.. 27
2.14: Control Box Schematic 29
2.15: Control Box 30
3.1: Full Configuration, Open Loop 32
3.2: Full Configuration, Closed Loop 39
4.1: Full Response, Configuration 1 46
4.2: Full Response, Separated State Variables.................. 47
4.3: Root Locus Plots for X1 through X4 48
4.4: Root Locus Plots for X% through X8 48
4.5: Transfer Function; RECS Testbed (DSA) 49
4.6: Transfer Function; Theoretical Model 50
4.7: Transfer Function Comparison 51
5.1: Configuration 1: Systems ID Mode 52

5.2:	Regulation	54
5.3:	Tracking	54
5.4:	Configuration 2: Input Disturbance Rejection	55
A.1:	RECS Schmatic	64
A.2:	Control Box Schmatic	65
A.3:	Connectors	67
B.1:	Interactive Animation Window	75
B.2:	RECS Superblock	76
B.3:	Derivative Superblock	77
B.4:	Sensors Superblock	78
B.5:	Control Superblock	79
B.6:	Disturb_Input Superblock	80
B.7:	Drive_Input Superblock	81
B.8:	Analog_Out Superblock	82
D.1:	Spring/Damper Mount	107
D.2:	Spring Holder	108
D.3:	Power Supply Stand	108
D.4:	Endpiece	109

LIST OF TABLES:

2.1: Servo Amp Power Requirements 13

2.2: Power Supply Power Sources 15

2.3: Servo Motor Specifications 17

2.4: RMA Specifications 25

4.1: Parameters of Model Fit 52

A.1: Wiring List 61

A.2: AC100 Connections 62

A.3: Unconnected Pins List 63

A.4: Control Box Components 66

A.5: AC100, IP-QUAD Connections 68

A.6: AC100, IP-DAC6 Connections 69

C.1: Associated Matlab files 83

ACKNOWLEDGMENTS

I state the obvious when I say that without my advisor, Dr. Juris Vagners, I would not have been able to do this, but it is true beyond the need for project supervision. From the beginning, his combination of helpfulness and professionalism let me know that I was very fortunate for the opportunity to study under him. He balanced essential direction with a chance to develop independent problem-solving skills. He has given me a desire for a deeper understanding of the control systems world; however, he was not the lone influence in my great educational experience. The rest of the UW control systems group is also gifted in intellect and humor. Thank you especially Doug, Anhtuan, Dave, Kalev, and Brian for the plethora of personality and the wealth of advice that you poured in the mix.

In addition, I want to thank Col. Jack Johnson. Throughout my entire educational tour he taught by example about leadership of the most honorable kind. He will not get a medal for the hours he spent to help launch my career, but his model of service leadership will be emulated in all of us who have had the privilege to serve under his command.

Finally, my inspiration in the lab came from sources beyond its walls. I deflect full praise to my Lord and Savior, Jesus Christ, for the continued blessing and guidance that has brought me thus far. Thank you also to Ryland and Sheryl Hardman, my parents and preeminent advisors.

Chapter 1

INTRODUCTION

1.1 Motivation

This hardware emulation testbed is, at its core, a fundamental rectilinear spring/mass/damper system. Rectilinear elastically-coupled systems are very traditional for academic interests; consisting simply of a disturbance input to a series of aligned masses joined by elastic and/or inelastic bonds (e.g. springs and dampers; See Figure 1.1). The ideal system would be one in which the masses have full range of motion in the axis of the disturbance, friction is nonexistent or linear, and all force transmissions are linear.

This ideal case is often used as a physical analog to some classes of differential equations. The system can also be related to matrices of system dynamics and their eigenfunctions [e.g. 3,6]. Physical systems are valuable to both validate the theory and to explore the nonlinearities in 'real world' coupled systems. They are also useful to validate the mathematical models presented in dynamics or structures course work.

Another area of interest is the frequency response of physical systems. With a disturbance source of suitable bandwidth, the system can be used to explore topics related to resonance, physical modes, and constructive/destructive interference.

A system with the same basic structure is useful in the field of control systems. This more complex scenario requires the same physical system with some modifications in the software. The mass position sensors can be fed back to alter the motor input for a desired position output. This forms a closed loop system that can

be implemented using a host of controller achitectures and provides easy performance evaluation.

Yet another system with a similar basic structure is useful in the field of control systems. This more complex scenario requires that the physical system already mentioned be modified by using the optical encoding sensors and some computer interface software to drive an installed reaction mass actuator (RMA). The RMA is mounted midsystem and coupled to the system on both sides (See Figure 1.1). In this way, the system becomes the physical representation of a general n^{th}-order system confined to rectilinear displacement. In this form the motor becomes the source of both a constant input and deterministic (or stochastic) disturbance. The RMA is used as the control actuator to dampen disturbances while transmitting desired input signals. In this scenario the inputs can be multiple frequency signals convolved in the computer and fed into a single actuator.

If a system is designed from the beginning as a multi-role testbed, it provides much more utility for both student experimentation and academic demonstration.

1.2 Purpose

The primary impetuous for the creation of a Rectilinear Elastically-Coupled System (RECS) was to give the University of Washington Controls and Robotics Systems Laboratory (CRSL) a testbed that was versatile enough to be used in all of the above-mentioned applications. The three configurations mentioned fall into three areas of control study: system identification, regulation/tracking, and input distur-

bance rejection. All three areas of study can benefit from convenient hardware-in-the-loop simulations.

A good RECS testbed should fulfill its first function as a spring/mass/damper system with a large amount of variability in all parameters. It should also be capable of variable-order configurations in order to provide a wide range of systems for identification. In addition to the open loop features, the closed loop form of the testbed should provide dynamic simulation during feedback control study and development. In this second and third configuration, the nonlinearities of real-world tests (such as static friction, coulomb friction, backlash, and kinematic nonlinearities) should be minimized and accounted for (or isolated and accented) as desired for experimentation. The conceptual layouts of such a testbed are shown in Figure 1.1 where each X_n

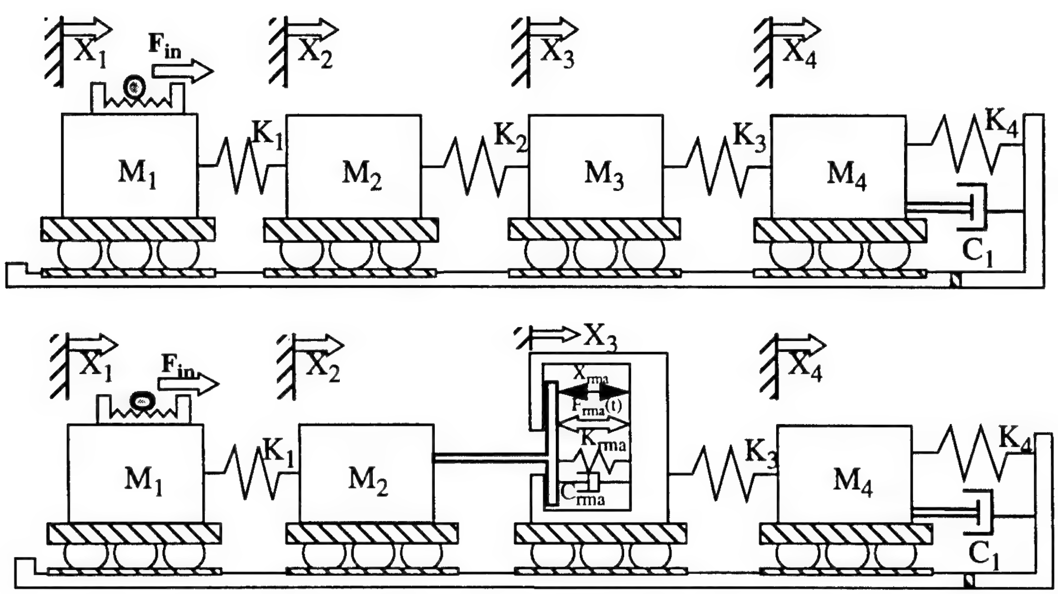


Figure 1.1: Full Configurations for Both Forms

represents the linear position of each mass M_n and the K_n terms represent spring con-

stants. As it can be seen, the third mass in the second configuration has been replaced by a reaction mass actuator. An optional dashpot damper is shown connected to mass M_4 of both configurations. The present configuration does not have this feature, but the final spring mount accommodates its addition.

A picture of the final product, a *Reconfigurable Hardware Testbed for Elastically-Coupled Systems* is shown in Figure 1.2. This device is now located in the UW

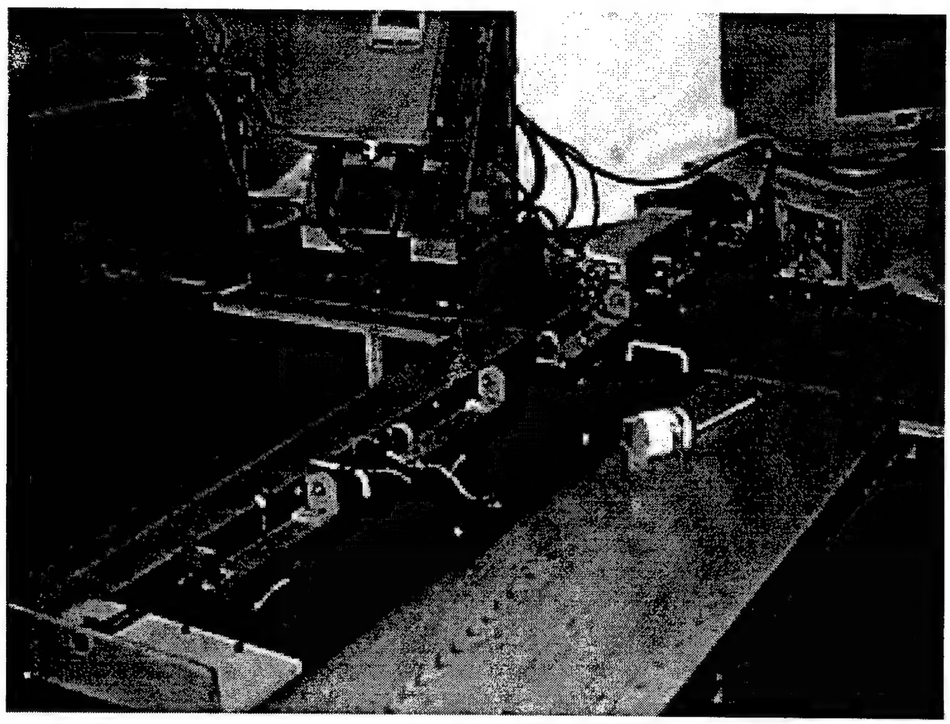


Figure 1.2: UW CRSL RECS Testbed

CRSL. The open loop configuration is intended to primarily complement undergraduate physics and structures coursework. The closed loop configurations can be utilized for upper-level undergraduate or graduate-level controls courses. The testbed was assembled with the requirement for reconfiguration in mind and therefore is

made to be easily modified to fit a plethora of mathematical models. All parameters for the mathematical models discussed in chapter 3 can be modified in hardware with only a screwdriver and an allen wrench. Furthermore, the masses can be increased on the fly using any standard centrally-bored laboratory weights.

1.3 Previous Work

As similar systems have been studied since even before Newton, the system's behavior is basically conquered ground. There are, however, some interesting recent developments in the area of educational tools. Many Matlab and C files are available to run simulations on PC or Unix systems. Several such programs that were written at the University of Washington are provided in Appendix C. A company by the name of Educational Control Products (ecp) has done much work in developing hardware systems to demonstrate principles of physics and control theory. As yet, they do not have any systems that parallel the RECS testbed in its third configuration form.

I would be in error if I did not mention the work done by a previous graduate student, Clint Schneider, who began this project. Though he completed his degree before I arrived, he still made a return visit to explain the vision of this project. One of the simulation Matlab codes in Appendix C was written by him and is noted as such.

1.4 Thesis Outline

The basic components of the RECS testbed are expounded upon in Chapter 2. Information in each section includes a description of the component's form and

function. This is written with the perspective that the chapter will leave the reader prepared to maintain, use, modify, and replace any significant components. Chapter 3 walks through the process of modeling RECS in the open and closed loop configurations. For completeness, the full order configuration is used in both cases, but lower order configuration models are easily determined from the given system matrices. In Chapter 4 computer simulations are compared to the real testbed. The simulations are executed in Matlab and the real system frequency analysis is derived from tests run with the Digital signal Analyzer in the UW CRSL. The simulation code can be perused in Appendix C. In Chapter 5, the foundational work of controller development and the information needed to interface with RECS is given. Chapter 6 is some concluding discussion. Both Chapters 5 and 6 should be helpful in getting future projects on the right track.

Appendix A gives all of the meticulous wiring details for the testbed including any information that may be needed to change sensors or actuators. Appendix B discusses the AC100 side of the testbed and gives a checklist for system start-up. **A separate copy of the Operating Checklist is with the testbed and should be strictly adhered to for safety and proper operation.** Appendix C is the Matlab simulation code, and Appendix D gives the specification diagrams of the testbed parts that were machined in-house at the UW AA Department's machine shop.

Chapter 2

TESTBED DESCRIPTION

2.1 Testbed Layout

The project pictured in Figure 1.2 is represented in diagram form in Figure 2.1 with attention given to the individual devices and interconnections. Each component is explained in the following sections.

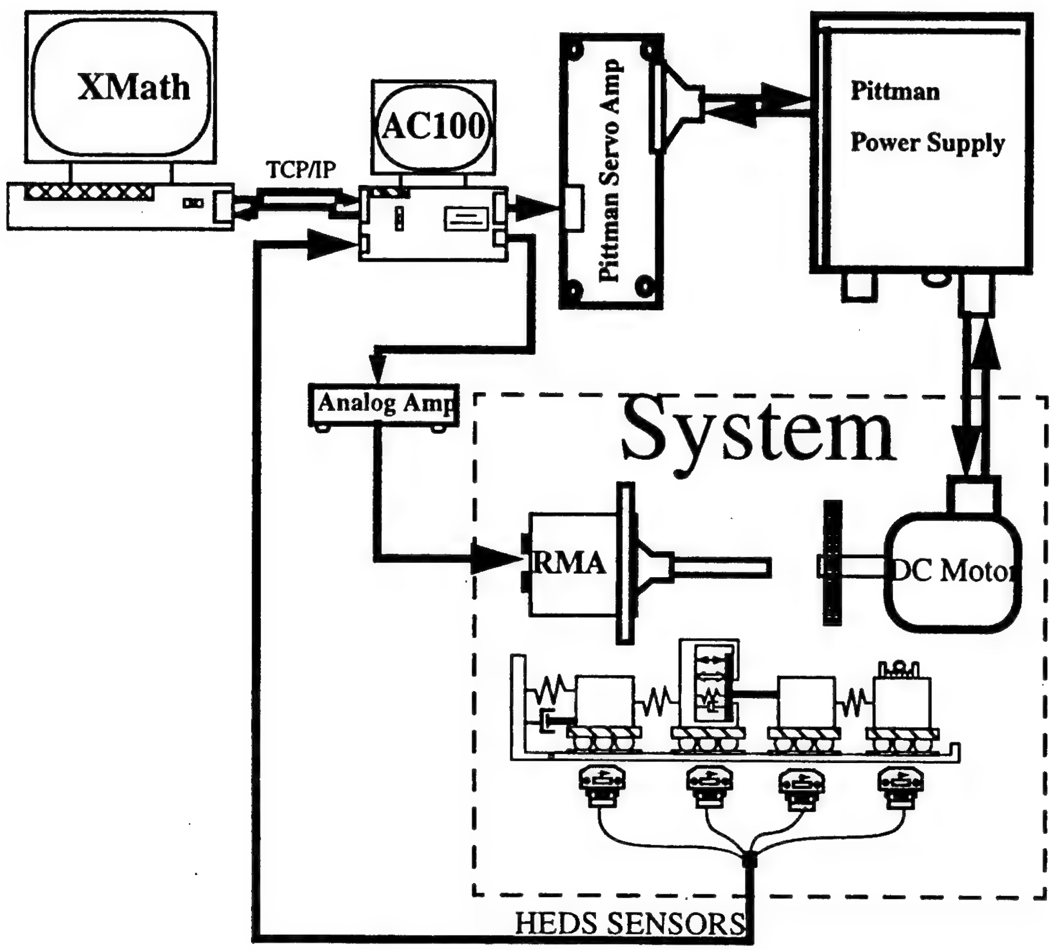


Figure 2.1: Block Diagram of RECS Testbed

2.2 Xmath Software

This project was built with the intention that it would be interfaced with controllers designed in some appropriate software package. Currently the UW CRSL uses Xmath/SystemBuild (v. 4.0) from Integrated Systems, Inc. This replaces the MATRIXx software. This software package, coupled with the AC100/C30 Real-Time Control System hardware, provides a flexible analysis and simulation environment. This testbed was made compatible for use with the existing setup. Appendix B shows the Systemblock construction and other useful software development information.

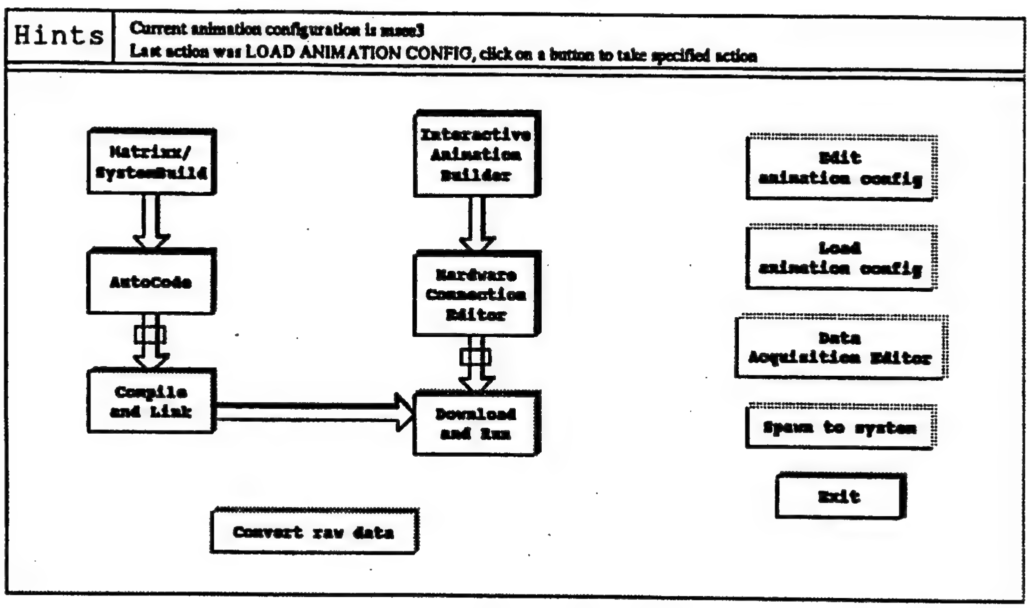


Figure 2.2: Xmath GUI in RealSim

Xmath is tailored for control systems work and runs on an X-Terminal/Sun Workstation in a UNIX operation system. Xmath is matrix based, similar to Matlab, but with Computer Aided Engineering (CAE) graphical, interactive editing capabil-

ities. The software can also operate in real-time to display output information and give commands to the system. The user receives this information via Graphical User Interface (GUI) displays in RealSim (a part of the Xmath package). One such display for simulation development is pictured in Figure 2.2. Real time interfacing can be developed visually with SystemBuild's interactive graphical block diagram editor. These, in turn, can be changed to executable code by using AutoCode. Xmath can also operate in either continuous or discrete mode and controllers designed in one mode can be converted. All of these options, and the flexibility of the RECS testbed, give the developer a powerful development tool [7,8].

2.3 AC100 system

The AC100 Real-Time Control System from Integrated Systems, Inc., (ISI) provides the flexible interface to complete the test environment. The AC100 hardware in use at the UW CRSL is a model C30 and is installed in a 486DX2/66MHz PC. It includes a Texas Instruments Digital Signal Processor TMS320C30 and a DSPFLEX carrier board. Also, the current setup has four IP module cards. These communicate through the DSPFLEX board.

The Quadrature Encoder Module (IP-QUAD) can receive input through four individually-configured channels. It processes data with four high speed 24-bit counters. There are also a variety of software modifications that can concatenate or group the channels. Another valuable feature is that individual channels of the IP-QUAD card can work in Absolute Position Mode or Velocity Mode and accommo-

dates linear or rotary sensors. This gives a great amount of flexibility to the types of sensors that can be utilized [7]. In the current RECS testbed setup, the AC100 receives input from four linear optical encoders. These report the position of the (up to) four linear sliders. The testbed is connected to the PC with AC100 via a 50-conductor IDC connector on the control box (see Appendix A). The remaining four states (the respective velocities) are computed internally.

The digital-to-analog module (IP-DAC6) is useful for output control signals such as the motor control. It has six 12-bit channels. Each channel is configurable to one of ten output ranges and is referenced to a single high precision reference. Maximum output current is +/- 2.5 mA [8]. In the current RECS testbed setup, the AC100 uses the IP-DAC6 to output two analog signals. Both are on the +/- 10 v range. One is the drive signal for the motor, and one is a control signal to the Reaction Mass Actuator (RMA).

Figure 2.3 shows the interaction between the computers. When a real-time simulation is under way, the PC-based AC100 is independent of the Sun Workstation. The simulation can be observed and affected, however, through the graphical user interface of the Xmath software. This information is transferred between the two computers via a an Ethernet connection with TCP/IP protocol [7,8].

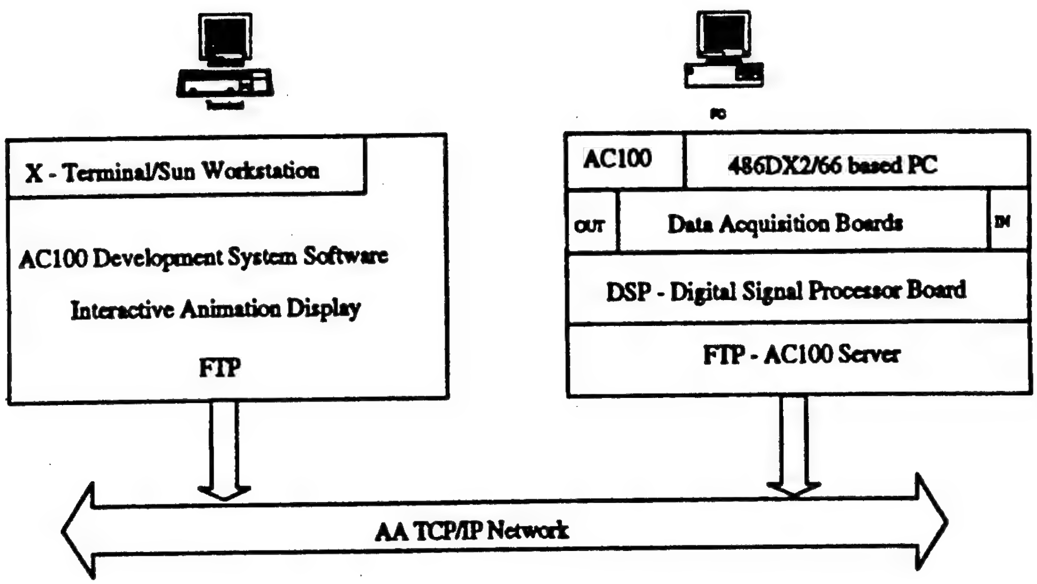


Figure 2.3: Computer Setup

2.4 Servo Amplifier

The motor drive signal from the AC100 goes into a Pittman servo amplifier. The servo amplifier used with the RECS testbed is packed with many great features which will be covered here in detail. This is because information on specific components is hard to obtain. In fact, this project was a very difficult system identification problem in an unorthodox sense. The information listed below was primarily obtained from the Pittman product bulletins (see References), but the servo amplifier, motor, and power supply were all purchased some time ago and were willed to this project a bit scarce on necessary details. After much research and some testing, the following information has been validated and recorded here for posterity. It would behoove the future user to refer to this section often.

The RECS testbed employs the Pittman ASM# 304-014-211 which is made to drive the Pittman ELCOM 3-phase motor with up to 500 watts of power. It operates in 4-quadrant mode and 6-step or Lo-Cog 12-step commutation. The schematic of this specific model is shown in Figure 2.4 and its picture is in Figure 2.5. This model is set to operate in Pittman's patented Lo-Cog 12-step commutation, fit into a double-width VME rack, and connect via a mounted plug-in Schroff connector at P1. All three of these parameters were selected from several options but modification can only be done by the manufacturer.

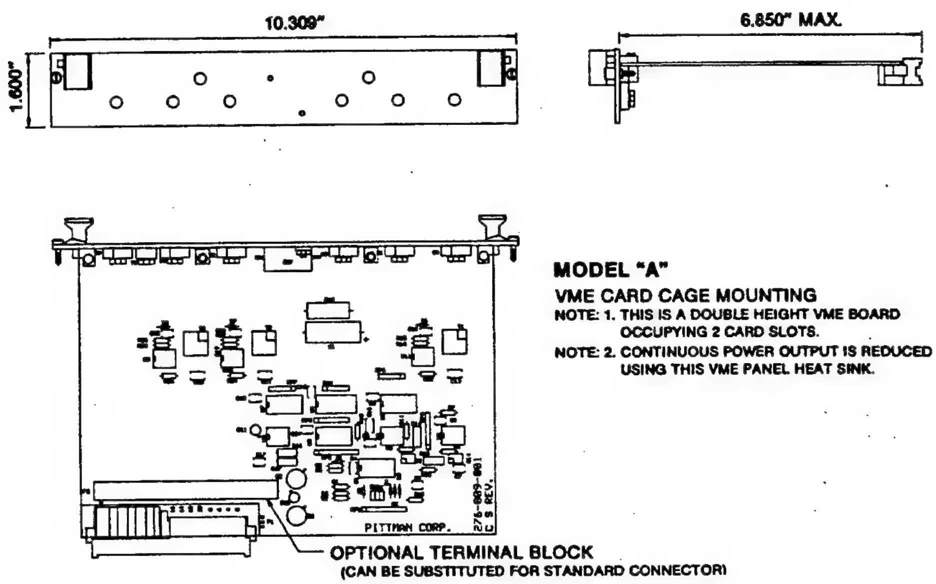


Figure 2.4: Servo Amplifier Schematic

In effect this model acts as a voltage controlled current source. It receives a +/-10V input signal and outputs a proportional current (adjustable from 0.5 V/A to 1.5 V/A; default is 0.75 V/A) to drive the motor at a proportional torque. For specific wir-

ing connections, consult Appendix A, but note, the card must have jumper J2 **(and only J2)** in place. The jumper positions can be found on the lower right side of the card as shown here in Figure 2.5.

The servo amplifier requires the following DC supply voltages:

Table 2.1: Servo Amp Power Requirements

Voltage	@ Amps
+12	0.25
-12	0.1
+5	0.35
0-70	motor req.s

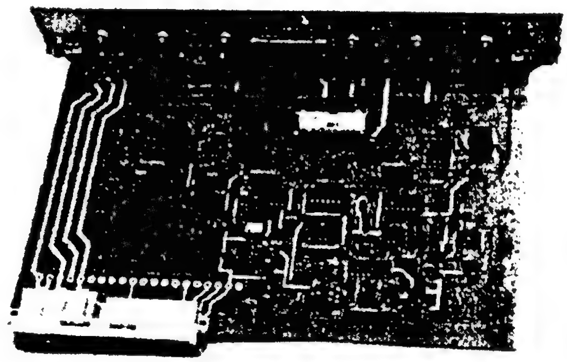


Figure 2.5: Servo Amplifier Card

The drive transistors are N-channel MOSFET technology and the servo amplifier is connected for thermal overload protection to protect the transistors. Also, the servo amplifier has a small signal bandwidth of 2kHz and a pulse width modulation frequency of 20kHz.

Two features that are not being used in the current configuration are the hall sensor outputs from the motor and the current monitoring feature. The hall sensors work off of the hall effect which says that a conductive element subject to a voltage in one axis and a perpendicular magnetic field will generate a voltage in the axis perpendicular to both the original voltage and the field [2]. The outputs of the installed hall sensors connect directly to the servo amplifier to close the loop of its control, but they could also be monitored externally to examine motor speed. The current monitoring could also be used externally. Since the drive current is proportional to torque which

is proportional to acceleration, the current output could be used as a state variable of the system. These options are all available for future use, but were unnecessary in the present configuration.

2.5 Power Supply

The muscle behind the servo amplifier's motor control comes from a Pittman 500 Watt linear unregulated DC power supply. This line of power supplies are made to

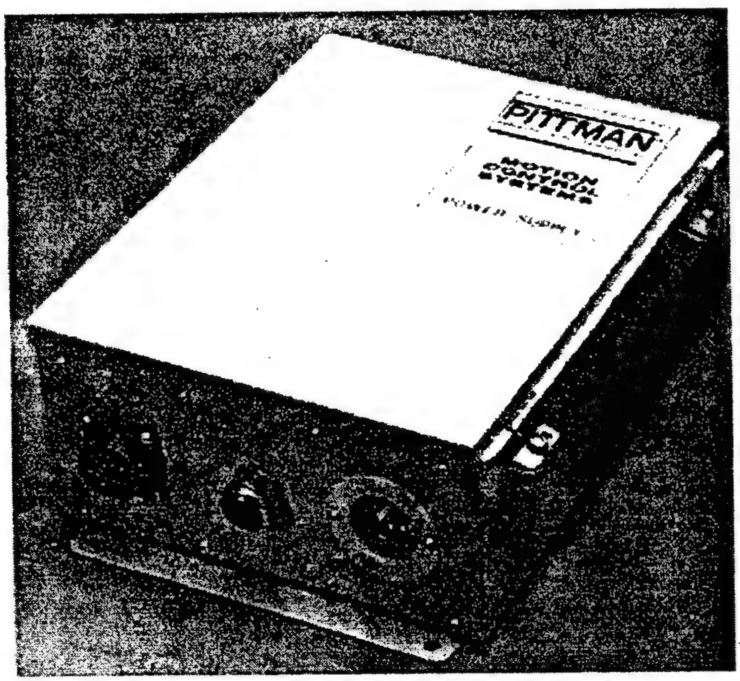


Figure 2.6: Power Supply in NEMA 12 Enclosure with AMP Connector

compliment the Pittman servo amplifier and motor being used on RECS. Specifically, the one used is Model #304-003-001. This is power supply, pictured in Figure 2.6, is enclosed in a NEMA 12 metal box with a Hubbell connector for its required 120 VAC, 60 Hz power input and an AMP connector for output. Interlock protection dis-

ables the output to the motor if the power supply and the servo amplifier become disconnected. The specific outputs are:

Table 2.2: Power Supply Power Sources

Voltage	@ Amps
+5	3
+12	.3
-12	.3
33	motor req.s

As it can be seen by comparing Tables 2.1 and 2.2, this power supply is adequate for the present control arrangement. The logic outputs are regulated and provide less than 200 mV ripple. The power supply output is unregulated and has an impedance of 1.6 ohms. Besides providing power for the motor and servo amplifier, this power supply has two separate power outputs. One is being used to proved the +5V necessary for the optical encoders, and the other (+12V and -12V) is unused at this time.

2.6 Brushless DC Servo Motor

The DC motor mounted on the RECS testbed completes the Pittman servo trio. It is an ELCOM 5112, WDG #3, 3-phase motor with the Lo-Cog 12-step drive option. This is Model #304-028-0504 and is drawn in Figure 2.7. In this drawing it is referred to by its discontinued manufacture part #5272B864.

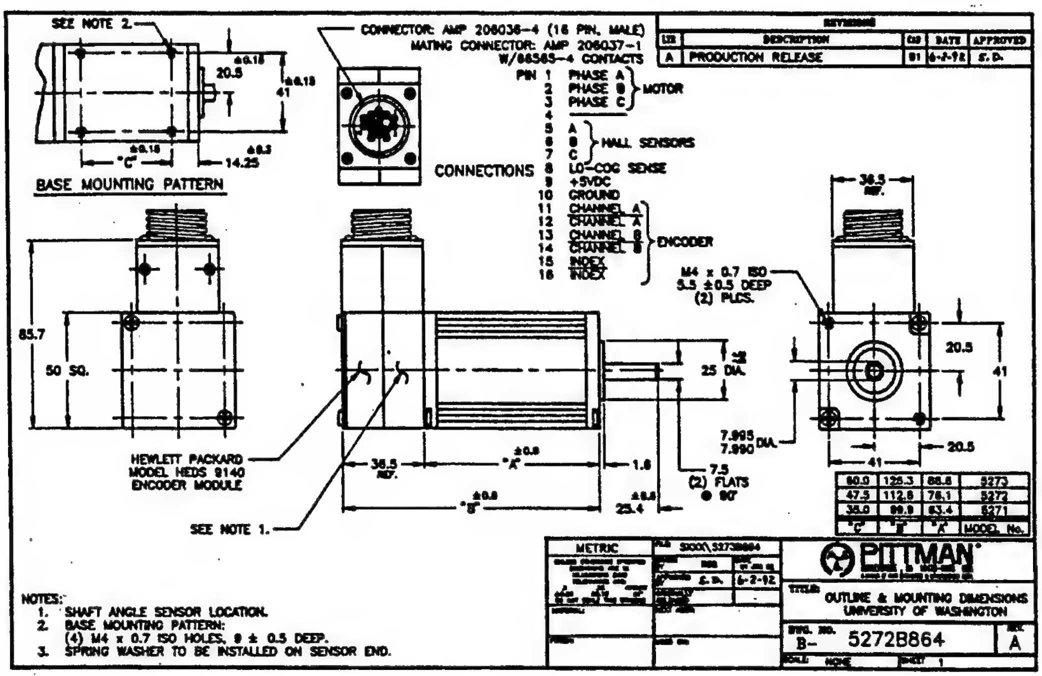


Figure 2.7: DC Motor Drawing

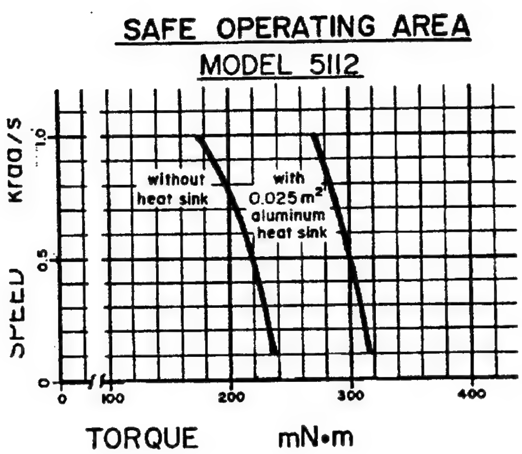


Figure 2.8: Motor Envelope

This motor has a safe operating range shown in Figure 2.8 and the specifications shown in Table 2.3. The motor being used on the testbed has been tested and found acceptable within the tolerances of the listed parameters. If precise values are required consult Pittman or perform the motor parameter tests described in [7].

Having the Lo-Cog option gives theoretical commutation torque ripples as low as 3.4%. Also, the RECS motor has the rotary optical encoder installed but not currently

in use. The rotor structure of this motor allows rotational speeds up to 25,000 RPMs, but again, **check Figure 2.8 for safety** [9].

Table 2.3: Servo Motor Specifications

Parameter	Units	Symbol	Value
Damping Constant (KtKe/Rt)	N x m/(rad/s)	K_d	4.15x10^-3
Motor Constant (kt/Rt^(1/2)	N x m/W$^{1/2}$	K_m	64.4x10^-3
Mech. Time Const. (J/Kd)	ms	τ_m	6.50
Elec. Time Const. (L/Rt)	ms	τ_e	0.323
Moment of Inertia	kg x m^2	J_a	26.9x10^-6
Viscous Damping	N x m/(rad/s)	D_v	15 x 10^-6
Friction Torque	N x m	T_f	3.7x10^-3
Motor Mass	kg	M	0.80
Thermal Time Constant	min	τ_{th}	19
Thermal Impedance (WDG-ambient)	^{o}C/W	R_{th}	3.0
Max WDG Temp.	^{o}C	θ_{mx}	155
Torque Constant	N x m/A	K_t	94.0 x 10^-3
Back emf Constant	V/(rad/s)	K_t	94.0 x10^-3
Stator Resistance	Ω	R_a	2.13
Stator Inductance	mH	L_a	0.686

Some analysis is given below to expand on the meaning and application of the stated parameters. Also, their use in the motor selection for the RECS testbed is justified. This analysis is necessary for correct modeling of the transfer function from drive signal to input force, but is unnecessary if the components are given or the above parameters are known to be correct. To begin with, a motor can be mathemat-

ically modeled by either of the following differential equations:

$$v_a(t) = R_a i_a + L_a \frac{di}{dt} + K\omega$$

$$Kt \cdot i_a = J_a \frac{d\omega}{dt} + \mu(\omega) + T_L(t) \tag{2-1}$$

where the parameters are explained in Figure 2.9. Note that K_t is referred to torque constant and back emf constant in Table 2.3 and Figure 2.9. Simple unit conversions will show that all of these terms are equivalent in any coordinated set of units such as the metric system [2,5].

V_a = motor input voltage (V)
i_a = motor armature current (A)
R_a = motor armature resistance (Ω)
R_s = amplifier resistance (Ω)
$R_m = R_a + R_s$ = total resistance (Ω)
L_a = motor armature inductance (H)
K_t = torque constant (N-m/A)
ω = rotational speed of motor shaft (rad/s)
J_m = rotor inertia (kg-m^2)
$\mu(\omega)$ = motor friction (N-m)
T_L = load torque (N-m)

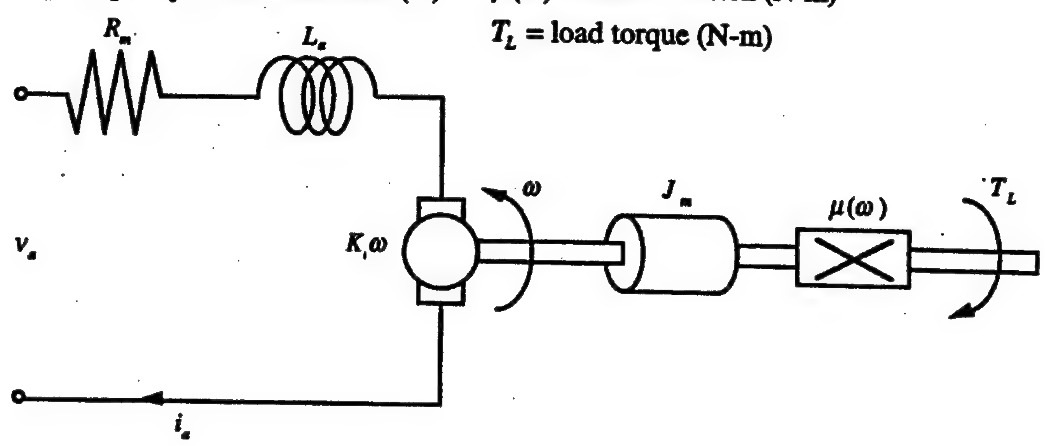


Figure 2.9: Motor model

A good initial model for the motor friction $\mu(\omega)$ is

$$\mu(\omega) = D_F \cdot \omega + T_F \qquad (2\text{-}2)$$

The equation for calculating the force F from the motor is

$$T_L = (F \cdot r) \qquad (2\text{-}3)$$

where r is the gear's effective radius. Thus, from Eqn.s 1-1 through 1-3 we find that the motor speed is proportional to the input voltage. More importantly, drive current is directly proportional to the torque which is directly proportional to the RECS testbed input force. After combining the servo amplifier and the motor in the present configuration we find that the input is a +/- 10v signal which ultimately yields a proportional force. Thus, we are in current control mode.

From Figure 2.8 the maximum torque is found to be about $T_L = 0.3$ Newton-meters and the anti-backlash gear has an effective radius of about r = 1.5 cm (full US diameter d = 1.25", so full radius, r = 1.5875 cm). This translates into a maximum possible input disturbance force of $F_{max} = 20N$. This met the requirements of RECS nicely. This is also used in spring selection which is covered in a later section.

Another useful piece of information is the transfer function from input signal magnitude (in volts) to output force (in Newtons). We will define the amplification factor between signal voltage v_s and servo amp drive current i_a as a constant set to its default value of 0.75 V/A. (Remember that this value is adjustable in the servo amp over the range from 0.5 to 1.5 V/A.) Also, the moment of inertia term J is now the

combined moment of inertia of the motor and gear $J = J_a + J_g$. Combining the information in the previous paragraph with Eqn.s 1-1 through 1-3 and expressing rotation in terms of θ (rad/s) we get:

$$K(0.75v_s) = (J_a + J_g)\ddot{\theta} + (D_F\dot{\theta} + T_F) + T_L \tag{2-4}$$

This is the complete equation, but Table 2.3 shows that J_a a and D_F are two or three orders of magnitude smaller than the other terms. For most application, this makes them negligible. Examining the gear shows that J_g is

$$J_g = \frac{1}{2}M({r_1}^2 + {r_2}^2) \tag{2-5}$$

by standard moment of inertia equations. The body weighs 50 grams and the internal bore is $r_1 = 8$ mm. This yields Jg = 7.9x10^-4. This is also small relative to the torques being generated. If we further constrain our equation to the relatively low velocities and accelerations that will be experienced in the RECS simulations, it becomes apparent that the first three terms on the right side of Equation 1-4 can be dropped. If we then group the torque due to the load with the torque due to internal friction the transfer function is seen to be

$$\frac{v_s}{T} = \frac{1}{0.75K_t} \approx 14.2\left[\frac{V}{N}\right] \tag{2-6}$$

This value can be useful for rough initial estimates and to obtain a comparison of overall efficiency.

2.7 Gearing

As mentioned before, the RECS testbed has a 1.25" anti-backlash gear mounted

to the motor. The drive system then connects to slider #1 with a rack and pinion arrangement as shown in Figure 2.10.

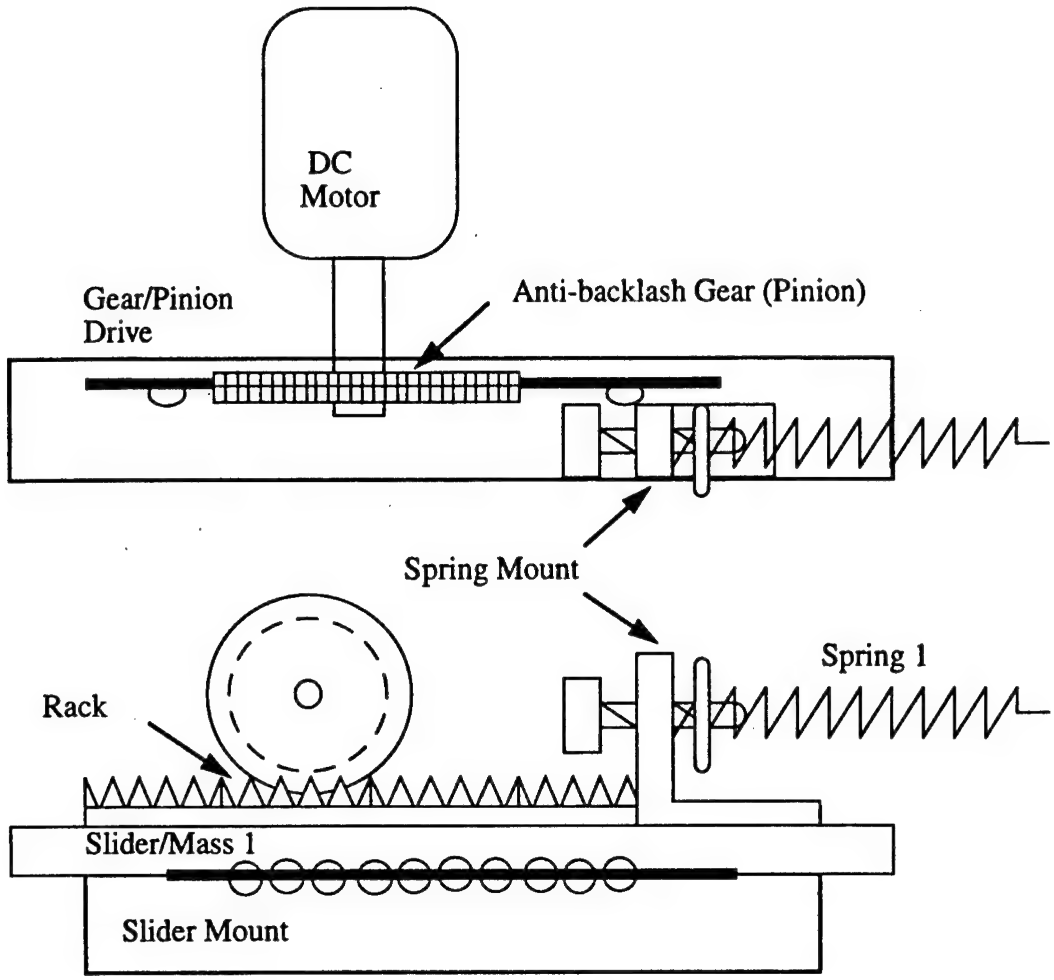


Figure 2.10: Top and Side View of Slider #1

This gear was selected in an effort to minimize the inherent nonlinearities of the drive system This gear is a stainless steel, 32 pitch, 64 teeth, set screw shaft mount, anti-backlash gear. The principle for this device is quite simple yet very effective. The device actually consists of two gears in juxtaposition with an internal spring-in-

duced force to hold them in staggered alignment. When the gear is interlocked with its mate, the gears are first twisted so that they interlock under spring tension. This tension absorbed most of the backlash phenomenon with surprisingly little increase in friction.

This is a definite recommendation for any future projects as a way to make very large reductions in system nonlinearity for a relatively small cost increase. The anti-backlash gear has reduced backlash in that part of the RECS testbed to below noticeable levels. No comparative test data is available between this and a traditional gearing arrangement because the traditional single gear was so obviously inferior.

One note is that this project required a gear with a 32 pitch, an 8.0 mm internal hole, and a set screw mount. No commercially available anti-backlash gear could be found to meet these exact specifications, so the present one was rebored in house without any noticeable reduction in effectiveness.

2.8 Sensors

Validity tests on the RECS testbed primarily center around the success of a given controller; how well it is able to control the position or velocity of a particular mass/slider in the system. Because of this, a precise set of sensors are required. The testbed currently uses four optical encoders mounted to each slider's base. These encoders produce a discrete quadrature output when a mylar codestrip passes between the emitter section and the detector section. The type of optical encoder currently being used is the Hewlett Packard linear optical incremental encoder module, HEDS 9200,

which is pictured in Figure 2.11. On this type, the emitter section consists of a LED with a polycarbonate lens to collimate the emissions. The detector section consists of multiple photodiodes and signal processing circuitry. The HEDS-9200 is able to detect a resolutions between 120 and 200 counts per inch (4.72 and 7.87 counts per mm).

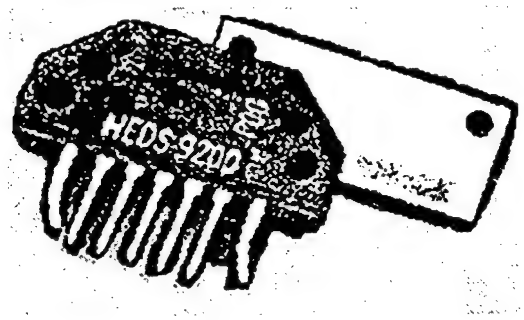


Figure 2.11: HEDS-9200 Optical Encoder

Running through each encoder is a 200 lpi (lines per inch) mylar strip that is mounted to the movable section of the slider. It is its motion that alternatively blocks and reveals the emitted light to the detection circuitry.

The HEDS-9200 requires a +5v supply and outputs two channels in quadrature. It also outputs the complement of both signals, but those are not necessary in the RECS electrical arrangement. The power and signal lines are connected to the control box through shielded coax 5-conductor cable.

2.9 Reaction Mass Actuator

The Reaction Mass Actuator (RMA) is a component that can be added for input disturbance rejection simulation. In this configuration, the servo motor functions as

the drive and disturbance input and the RMA is the control actuator. The RMA's used at the UW CRSL have been studied at length as they are used in several lab courses. The principle of operation is basically that of a speaker voice coil. The spindle is forced forward and back over it's range of motion. The movement is controlled by the forces created when the drive signal passes through a coil near a permanent magnet. The RMA's in the UW CRSL are Ling Dynamic Systems (LDS) V101 vibrators of the type shown in Figure 2.12 and specified in Table 2.4.

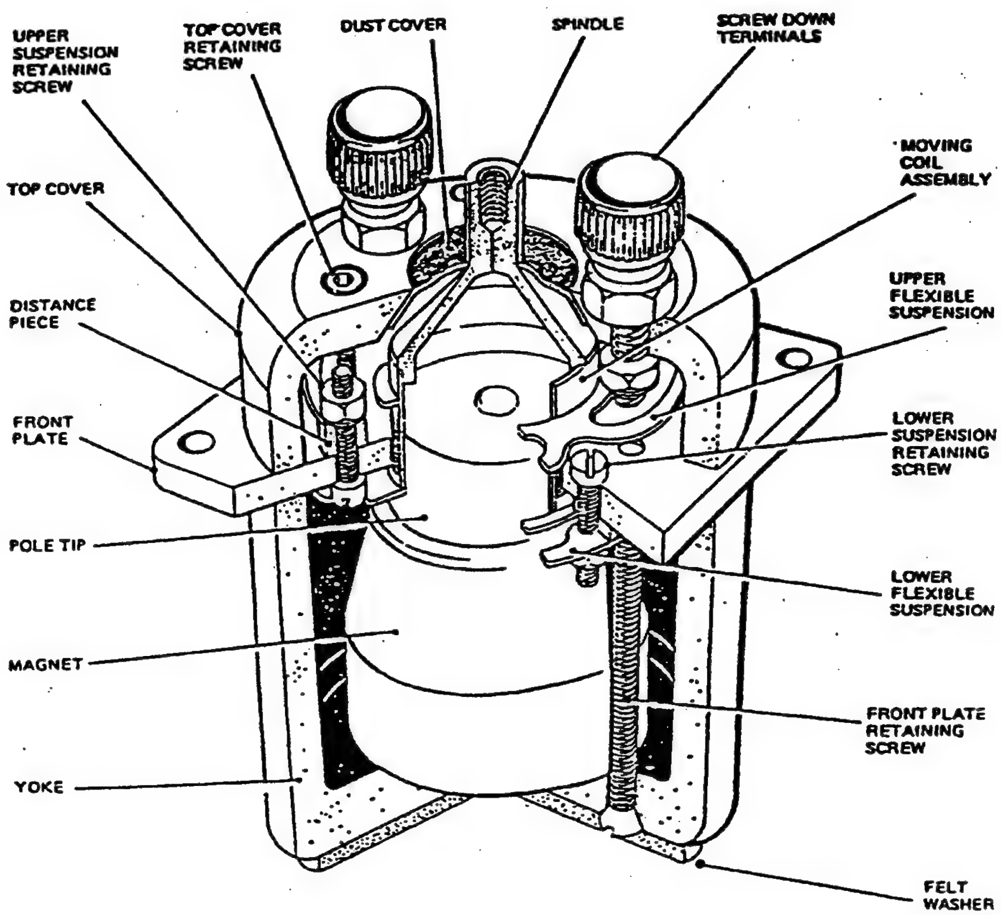


Figure 2.12: Reaction Mass Actuator

The RMA measures 89 x 65 x 65 mm (3.25 x 2.562 x 2.562 in) and weighs 0.91 kg (2.0 lb). When in dynamic mode, an effective mass of 0.0065 kg (0.0144 lb) must be added to the mass of the RMA to correctly model its effects on the system. This is because of the force applied. These parameters have been tested and have proven to be quite constant to over 1 kHz. At that point there are some significant swings which must be accounted for in operations performed there. [4]

Table 2.4: RMA Specifications

Specification	Metric	American
Max Sine Force peak	8.9 N	2 lbf
armature Resonance Freq.	12000 Hz	--
Useful Freq. Range	5-12000 Hz	--
Max Velocity Sine Peak	1.31 m/s	51.6 in/s
max Acceleration Sine Peak	1373 m/s^2	140 gn
Suspension axial stiffness	3.15 n/mm	18 lbf/in
Max Displacement pk-pk	2.5 mm	.1 in
Electrical requirement - Amplifier	0.09 kVA	--
Impedance (@500 Hz)	3 ohms	--

2.10 Ball Bearing Sliders

RECS moves on four 8" linear sliders. These free ball bearing track sliders are designed for very low friction. In their current state, however, they may fail to meet their optimum smooth ride because of maladjustment. Each slider has a line of adjustment screws that lie along one side (underneath each mylar encoder strip). With some tuning, better performance can be expected. These sliders are mounted to ad-

justable slider stands which can be moved to accommodate different spring lengths or to vary the distance between sliders. Also, the slider assemble accommodates a HEDS optical encoder. The sliders also have spring mounts or an RMA mount, and a mass mount. The free moving part of the slider and mass mount alone weigh about 332.0 g. This amount must be added to any explicitly added mass for correct modeling. Lab tests showed that the sliders could hold weight in excess of three kilograms without any significant decrease in performance.

Mounted to the top of the first slider is a 10", 32 pitch linear rack gear. This is the complement to the pinion gear that mounts on the motor shaft. This also must be added in separately.

2.11 Damper

No damper has been purchased at this time and significant tests can be performed without one. If one is installed in the future, however, it has been included in the mathematical model. Also, the mount on mass #4 is made to host both a spring and damper in juxtaposition.

2.12 Springs

The currently available springs are squared end ground 15 cm (6-inch) compression springs with outside diameters that can range from 2 to 5 cm. The springs have a linear deflection range of +/- 10 cm. The company, McMaster-Carr lists the springs by load rate. The springs available in the UW CRSL range from K = 5.5 to 22.0 with stiffness ratios [oz/in] of 48.0 to 8.0. In load rate that is 2.74 to 0.69 lpf-in. It is nec-

essary to express the desired spring value in all the above mentioned forms because supplying companies do not seem to have established any standardization.

RECS will host a very large range of spring stiffness (K), whether all matching or in any simultaneous combination; however, some prior thought should be put into spring selection. Factors that should be considered are desired mass displacement and resonant frequencies.

2.12.1 Mass Displacement

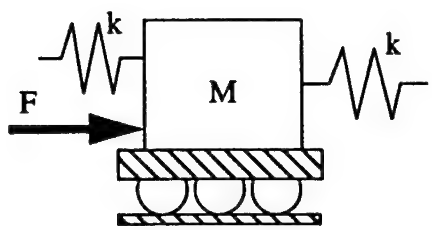


Figure 2.13: Spring Analysis

Figure 2.13 shows an individual slider mount. The maximum possible reactive spring force exerted by the springs will occur when both are at maximum displacement. At that point we want the force to be less than the maximum possible distürbance force in order to not exceed design limitations. As noted in Eqn. 2-4 the maximum force exertable by the brushless motor is $F_{max} = 20N$. So we see that: $Fk = 2kx \le 20N$. With a conversion factor of $k_{metric} \cdot (5.710 \times 10^{-3}) = k_{English}$ and a spring of length $l = 15\ cm$ the load rate is the constant multiplied by the total length of spring.

2.12.2 Resonant Frequency

Some open loop experiments may be particularly targeted towards examining the resonant frequencies of various RECS configurations. On the other hand, some lower frequency modes can become quite annoying if they are not the subject of interest. Either way, resonances should be a topic of interest in spring selection. From basic physics we know that $\omega = \sqrt{k/m}$ where m is the mass of the slider/mass component and k is the spring constant. The most likely scenario is that we will want to select a spring constant that accommodates a predetermined set of masses; either by moving the resonance beyond our area of interest or bringing it into the realizable picture.

2.13 Masses

The mass mounts were made to be very versatile in hosting mass configurations regardless of shape or size. The most secure method is to use laboratory-style round masses with a center hole and notch.

Values for the masses, springs, and dashpot were selected by first weighing the RMA which will become part of mass 2, and then selecting spring constants to adequately restrict motion so as not to overrun the linear slide max travel value of +/- 5 cm. The total mass of the RMA is $(M_{RMA} + M_{mount}) = 1.1217$ kg and the slider weighs 0.332 kg; therefore, all mass values were initially chosen to be 1.5 kg.

2.14 Control Box

This box actually contains no control elements but is the connection center for all the dependent components. The flexibility built into this area of the testbed was de-

signed primarily for changes to the entire sensor and control structure of the system. The schematic for this box is shown in Figure 2.14 on the next page and a full page version is shown in Appendix A. Note that the connectors designated by a small 'x' have been installed but are available for future expansion. For a complete listing of the components of this box, see Table A.1: in Appendix A.

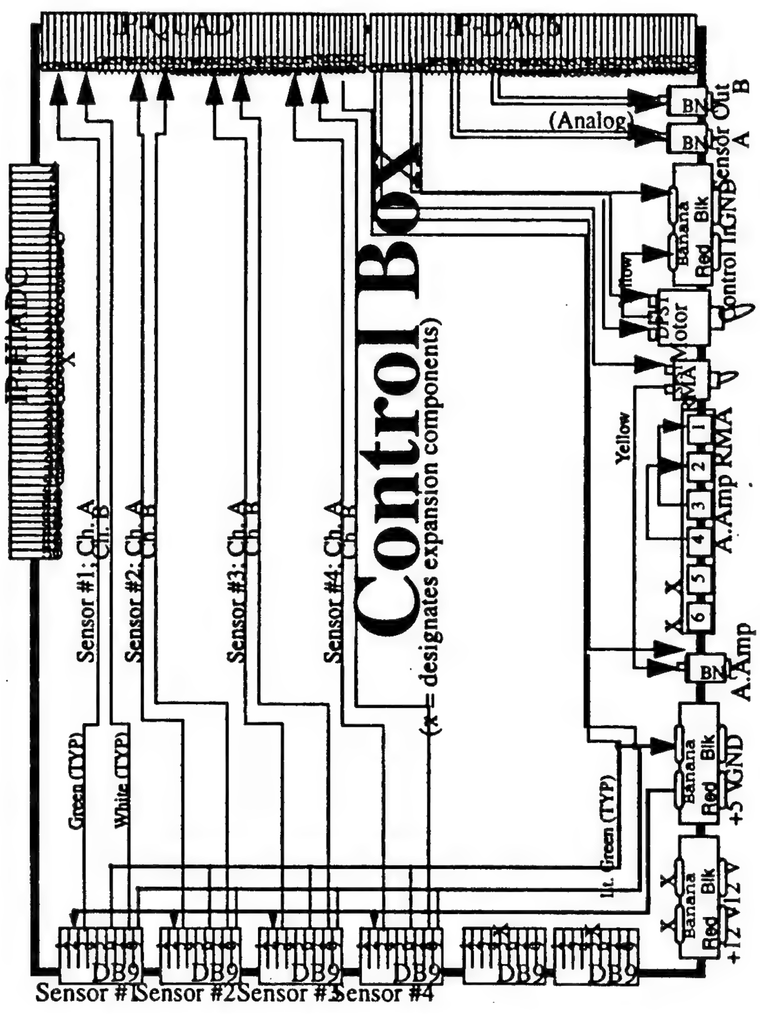


Figure 2.14: Control Box Schematic

The box features two expansion DB-9 connection slots for additional sensor in-

represents the linear position of each mass M_n and the K_n terms represent spring constants. As it can be seen, the third mass in the second configuration has been replaced by a reaction mass actuator. An optional dashpot damper is shown connected to mass M_4 of both configurations. The present configuration does not have this feature, but the final spring mount accommodates its addition.

A picture of the final product, a *Reconfigurable Hardware Testbed for Elastically-Coupled Systems* is shown in Figure 1.2. This device is now located in the UW

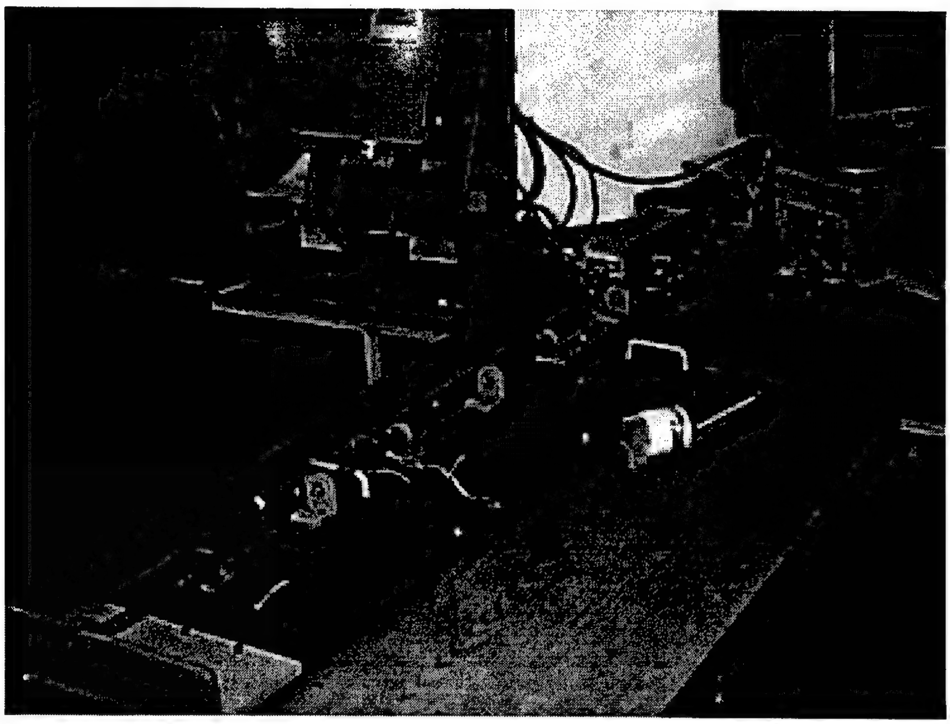


Figure 1.2: UW CRSL RECS Testbed

CRSL. The open loop configuration is intended to primarily complement undergraduate physics and structures coursework. The closed loop configurations can be utilized for upper-level undergraduate or graduate-level controls courses. The testbed

the RECS testbed. The remainder of the hardware was machined in-house at the University of Washington's Astro/Aeronautics Department Machine shop under the consultation of Dennis Peterson. Diagrams of these parts can be found in Appendix D. The noteworthy components are the endpiece, spring mount assembly, and the power supply unit stand. These bear explanation because of the need to be familiar with them before modifications to the testbed are performed.

Chapter 3

MODEL DEVELOPMENT

3.1 Modeling the RECS Testbed

Spring/mass/damper systems easily lend themselves to modeling via Newtonian physics, so it only made sense to mathematically model RECS using Newtonian notation. That is, through the exploitation of Newton's Second Law of Motion, $F = ma$, we get the dynamic equations for the systems shown in Figures 3.1 and 3.2.

3.2 Open Loop Configuration

While Figure 3.1 is labeled the open loop configuration, it also represents the configuration for tracking and regulation studies. In that case, $F_{in}(t)$ would be a control input from the developed controller and the sensor outputs would be used in closed loop feedback. In the open loop form, the system can be observed through the sensor output, as well as visually, but the sensor outputs do not affect what is input to the system.

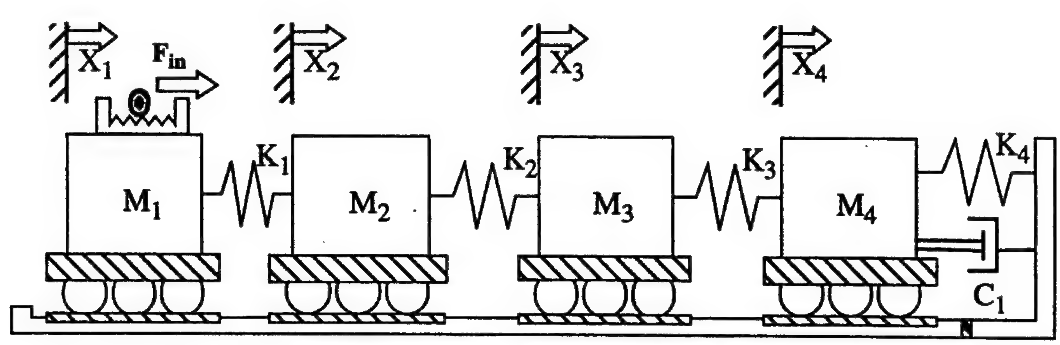


Figure 3.1: Full Configuration, Open Loop

3.2.1 Physical System

Four arbitrary masses $M_{1\text{-}4}$ are fastened atop ball bearing sliders that are interconnected with springs as shown in Figure 3.1. The respective spring constants are $K_{1\text{-}4}$. The motion of the corresponding masses is designated by $X_{1\text{-}4}$ and is relative to the designated origins. Input force F_{in} is applied with a brushless DC servo motor and is used to excite the system. The motor inputs the summation of all desired displacement forces. For example, if we wanted to drive the system with a force F_d with some simulated additional input disturbance F_x. We would sum those signals in the software and output to the motor $F_{in} = F_d + F_x$. The optional dashpot damper C_1 is an adjustable air pot. In the physical system, any combination and number of masses and springs (up to the number shown here) can be used. The end mount can be disconnected and moved forward for reduced-order systems. Only the models for the full order system follow, but some trivial reductions will yield any lower order mathematical and state-space models desired.

3.2.2 Mathematical Model Derivation

In the initial mathematical model, assume no friction, purely linear springs, and that all motion is confined to the region of linearity. Applying Newton's Second Law in superposition to the system described above yields:

- For mass #1

$$M_1\ddot{X}_1 = F_{in}(t) - K_1(X_1 - X_2) \tag{3-1}$$

- For mass #2

$$M_2\ddot{X}_2 = K_1(X_1 - X_2) - K_2(X_2 - X_3) \qquad (3\text{-}2)$$

- For mass #3

$$M_3\ddot{X}_3 = K_2(X_2 - X_3) - K_3(X_3 - X_4) \qquad (3\text{-}3)$$

- For mass #4

$$M_4\ddot{X}_4 = K_3(X_3 - X_4) - K_4(X_4) - C_1\dot{X}_4 \qquad (3\text{-}4)$$

3.2.3 State Space Formulation

3.2.3.1 Definitions

To put Equations 1-1 to 1-4 into state space form, the state variables are defined to be:

- x_1 = Position of Mass 1 (X_1) x_2 = Velocity of Mass 1
- x_3 = Position of Mass 2 (X_2) x_4 = Velocity of Mass 2
- x_5 = Position of Mass 3 (X_3) x_6 = Velocity of Mass 3
- x_7 = Position of Mass 4 (X_4) x_8 = Velocity of Mass 4

Then defining

$$\dot{\bar{x}} = A\bar{x} + Bu \qquad (3\text{-}5)$$

$$\bar{y} = C\bar{x} + Du \qquad (3\text{-}6)$$

and

$$u = F_{in} = F_d + F_x \qquad (3\text{-}7)$$

The matrix is derived from examining the parameters that are fed back from the system to the controller. The RECS testbed currently has the following:

- Encoder #1 = Position of M_1
- Encoder #2 = Position of M_2
- Encoder #3 = Position of M_3
- Encoder #4 = Position of M_4

Since the four fed back values correspond to the y_n terms, and since the positions and velocities of the respective masses correspond to state variables:

- $y_1 = x_1$
- $y_2 = dy_1/dt = x_2$
- $y_3 = x_3$
- $y_4 = dy_3/dt = x_4$
- $y_5 = x_5$
- $y_6 = dy_5/dt = x_6$
- $y_7 = x_7$
- $y_8 = dy_7/dt = x_8$

The odd-valued state variables, therefore, are obtained directly from the sensor outputs, but the even-valued state variables have to be determined through numerical differentiation in software.

3.2.3.2 State Matrices

With the above definitions, the state matrices become:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-K_1}{M_1} & 0 & \frac{K_1}{M_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{K_1}{M_2} & 0 & \frac{-(K_1+K_2)}{M_2} & 0 & \frac{K_2}{M_2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{K_2}{M_3} & 0 & \frac{-(K_2+K_3)}{M_3} & 0 & \frac{K_3}{M_3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \frac{K_3}{M_4} & 0 & \frac{-(K_3+K_4)}{M4} & \frac{-C_1}{M_4} \end{bmatrix}$$

and

$$B = \begin{bmatrix} 0 \\ \frac{1}{M_1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad ; C = \bar{I}_{8x8} ; D = \bar{O}_{8x1}$$

3.2.4 Model with Friction Compensation

3.2.4.1 Equations

The model derived above is theoretically correct. System response plots show that it is unstable without a sufficient damper C1, which was foreseeable. To more accurately represent a real system, friction components are added to each of the system equations. While friction has nonlinear characteristics, a significant improvement in our model can be made by simply adding velocity coefficients F_{μ} that represent the viscous friction of each linear slider [1]. Applying Newton's Second Law in superposition to the new system yields:

For mass #1

$$M_1\ddot{X}_1 = F_{in}(t) - K_1(X_1 - X_2) - F_{\mu 1}\dot{X}_1 \tag{3-8}$$

For mass #2

$$M_2\ddot{X}_2 = K_1(X_1 - X_2) - K_2(X_2 - X_3) - F_{\mu 2}\dot{X}_2 \tag{3-9}$$

For mass #3

$$M_3\ddot{X}_3 = K_2(X_2 - X_3) - K_3(X_3 - X_4) - F_{\mu 3}\dot{X}_3 \tag{3-10}$$

For mass #4

$$M_4\ddot{X}_4 = K_3(X_3 - X_4) - K_4(X_4) - (C_1 + F_{\mu 4})\dot{X}_4 \tag{3-11}$$

3.2.4.2 State Matrices

With all the previously mentioned definitions still apropos, the state matrices become:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-K_1}{M_1} & \frac{-F_{\mu 1}}{M_1} & \frac{K_1}{M_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{K_1}{M_2} & 0 & \frac{-(K_1+K_2)}{M_2} & \frac{-F_{\mu 2}}{M_2} & \frac{K_2}{M_2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{K_2}{M_3} & 0 & \frac{-(K_2+K_3)}{M_3} & \frac{-F_{\mu 3}}{M_3} & \frac{K_3}{M_3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \frac{K_3}{M_4} & 0 & \frac{-(K_3+K_4)}{M4} & \frac{-(C_1+F_{\mu 4})}{M_4} \end{bmatrix}$$

and B, C, and D are unchanged.

3.3 Closed Loop Configuration

While there are many applications for the RECS testbed in the configuration discussed so far, the testbed was designed to go further. The more complicated closed loop configuration seen in Figure 3.2 is related to the open loop configuration but has a Reaction Mass Actuator (RMA) replacing mass M2.

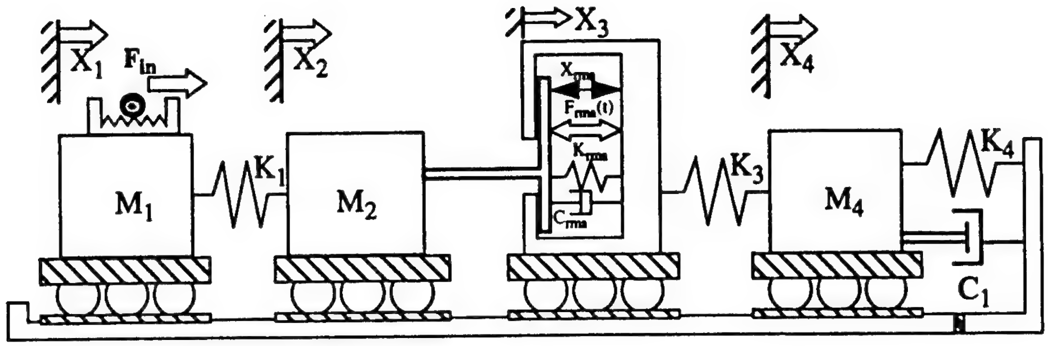


Figure 3.2: Full Configuration, Closed Loop

3.3.1 Physical System

The RMA works in a similar manner to an audio speaker. It produces a force by passing a varying electric current through a constant magnetic field. It is modeled as a force (F_{rma}) in parallel with a spring (K_{rma}) and a damper (C_{rma}) as shown in Figure 3.2. The RMA force is given by:

$$F_{rma}(t) = \frac{K_t}{R} V_{rma} \tag{3-12}$$

Where V_{rma} (volts) is the externally applied control signal, K_t (N/Amp) is the electromechanical force constant, and R (ohms) is the RMA internal resistance. K_{rma} and C_{rma} are properties of the voice coil material. Displacement distance X_{rma} indicates the magnitude and direction that the RMA stinger has shifted from neutral. By defining $X_{rma} = 0$ for the neutral position, a positive X_{rma} value signifies an expansion of the RMA stinger and a negative value signifies a retraction.

3.3.2 Mathematical Model Derivation

Once again, assuming no friction, purely linear springs, and all motion is confined to the region of linearity. Recall that the mass of the RMA must include its dynamic component. Applying Newton's Second Law in superposition to this system yields:

- For mass #1

$$M_1\ddot{X}_1 = F_{in}(t) - K_1(X_1 - X_2) \tag{3-13}$$

- For mass #2

$$M_2\ddot{X}_2 = K_1(X_1 - X_2) - K_{rma}(X_{rma}) - C_{rma}(\dot{X}_{rma}) - F_{rma} \tag{3-14}$$

- For mass #3

$$M_3\ddot{X}_3 = K_{rma}(X_{rma}) + C_{rma}(\dot{X}_{rma}) + F_{rma} - K_3(X_3 - X_4) \tag{3-15}$$

- For mass #4

$$M_4\ddot{X}_4 = K_3(X_3 - X_4) - K_4(X_4) - C_1\dot{X}_4 \tag{3-16}$$

Notice that the equations for mass #1 and #4 are identical in the two configurations. The added RMA simply acts as a new set of parameters inserted between two of the masses. This is shown in Figure 4.4 as a split in the control diagram's plant model.

3.3.3 State Space Formulation

3.3.3.1 Definitions

To put Equations 1-13 to 1-16 into state space form, the state variables are defined as stated earlier. Also, the new state variable X_{rma} can be replaced by another $(X_2 - X_3)$ in order to maintain a system description of order eight. Rewriting Equa-

tions 1-14 and 1-15 to reflect this substitution:

For mass #2

$$M_2\ddot{X}_2 = K_1(X_1 - X_2) - K_{rma}(X_2 - X_3) - C_{rma}(\dot{X}_2 - \dot{X}_3) - F_{rma} \qquad (3\text{-}17)$$

For mass #3

$$M_3\ddot{X}_3 = K_{rma}(X_2 - X_3) + C_{rma}(\dot{X}_2 - \dot{X}_3) + F_{rma} - K_3(X_3 - X_4) \qquad (3\text{-}18)$$

Then defining:

$$\bar{u} = \begin{bmatrix} F_{in} \\ F_{rma} \end{bmatrix} \qquad (3\text{-}19)$$

The RECS testbed currently has the sensor configuration stated in the open loop configuration section previously.

3.3.3.2 State Matrices

With the above definitions, the state matrices become:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-K_1}{M_1} & 0 & \frac{K_1}{M_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{K_1}{M_2} & 0 & \frac{-(K_1+K_{rma})}{M_2} & \frac{-C_{rma}}{M_2} & \frac{K_{rma}}{M_2} & \frac{C_{rma}}{M_2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{K_{rma}}{M_3} & \frac{C_{rma}}{M_3} & \frac{-(K_{rma}+K_3)}{M_3} & \frac{-C_{rma}}{M_2} & \frac{K_3}{M_3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \frac{K_3}{M_4} & 0 & \frac{-(K_3+K_4)}{M4} & \frac{-C_1}{M_4} \end{bmatrix}$$

and

$$B = \begin{bmatrix} 0 & 0 \\ \frac{1}{M_1} & 0 \\ 0 & 0 \\ 0 & -\frac{1}{M_2} \\ 0 & 0 \\ 0 & \frac{1}{M_3} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} ; C = \overline{I_{8x8}} ; D = \overline{O}_{8x2}$$

3.3.4 Model with Friction Compensation

In the same way as Configuration 1, the model derived above is theoretically correct. It is also unstable without a sufficient damper C1. Once again, to get a more

realistic model, linear viscous friction components are added to the equations. Applying Newton's Second Law in superposition to the new system yields:

For mass #1

$$M_1\ddot{X}_1 = F_{in}(t) - K_1(X_1 - X_2) - F_{\mu 1}\dot{X}_1 \tag{3-20}$$

For mass #2

$$M_2\ddot{X}_2 = K_1(X_1 - X_2) - K_{rma}(X_2 - X_3) - C_{rma}(\dot{X}_2 - \dot{X}_3) - F_{rma} - F_{\mu 2}\dot{X}_2 \tag{3-21}$$

For mass #3

$$M_3\ddot{X}_3 = K_{rma}(X_2 - X_3) + C_{rma}(\dot{X}_2 - \dot{X}_3) + F_{rma} - K_3(X_3 - X_4) - F_{\mu 3}\dot{X}_3 \tag{3-22}$$

For mass #4

$$M_4\ddot{X}_4 = K_3(X_3 - X_4) - K_4(X_4) - (C_1 + F_{\mu 4})\dot{X}_4 \tag{3-23}$$

3.3.5 State Matrices with Friction Compensation

With all the previously mentioned definitions still apropos, the matrices take on the following form:

A is an 8x8 with the following elements:

$$
\begin{matrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{-K_1}{M_1} & \frac{-F_{\mu 1}}{M_1} & \frac{K_1}{M_1} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\frac{K_1}{M_2} & 0 & \frac{-(K_1+K_{rma})}{M_2} & \frac{-(C_{rma}+F_{\mu 2})}{M_2} & \frac{K_{rma}}{M_2} & \frac{C_{rma}}{M_2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & \frac{K_{rma}}{M_3} & \frac{C_{rma}}{M_3} & \frac{-(K_{rma}+K_3)}{M_3} & \frac{-(C_{rma}+F_{\mu 3})}{M_3} & \frac{K_3}{M_3} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & \frac{K_3}{M_4} & 0 & \frac{-(K_3+K_4)}{M4} & \frac{-(C_1+F_{\mu 4})}{M_4}
\end{matrix}
$$

and with B, C, and D unchanged. These matrices, and those of section 1.3.5 are used in the Matlab simulations of chapter 4.

3.4 Nonlinearities

Most of the nonlinearities of the physical system can be predicted, but they must be examined in hardware-in-the-loop analysis for verification. Viscous friction has already been discussed and a simple attempt has been made to model it.

Some others known to exist are:

- Spring nonlinearities: K_{r1}, K_{r2}, K_{r3}, K_{r4}
- Sensor transfer functions: y_n/x_n
- RMA transfer function: V_t/F_{rma}

These form the possibilities for some interesting examination using the RECS test-bed.

Chapter 4

SIMULATION AND MODEL COMPARISON

4.1 Matlab simulations

The state space mathematical model developed in Chapter 3 can be easily put into a matrix-based computational software package (such as Matlab or Matrix_x) and given simulated inputs. Simulating in this way allows all aspects of the system to be precisely controlled. The following figures were done using Matlab 4.0. Values were selected for all necessary parameters from the manufacturer's specification sheets. Applicable Matlab files include: `c1sim.m`, `c2sim.m`, `config1.m`, `config2.m`, and `master.m`. These can all be found in Appendix C.

Figure 4.1 shows the entire output set for a complete system in configuration 1. This plot highlights the phase shifting that occurs as the disturbance propagates through the system.

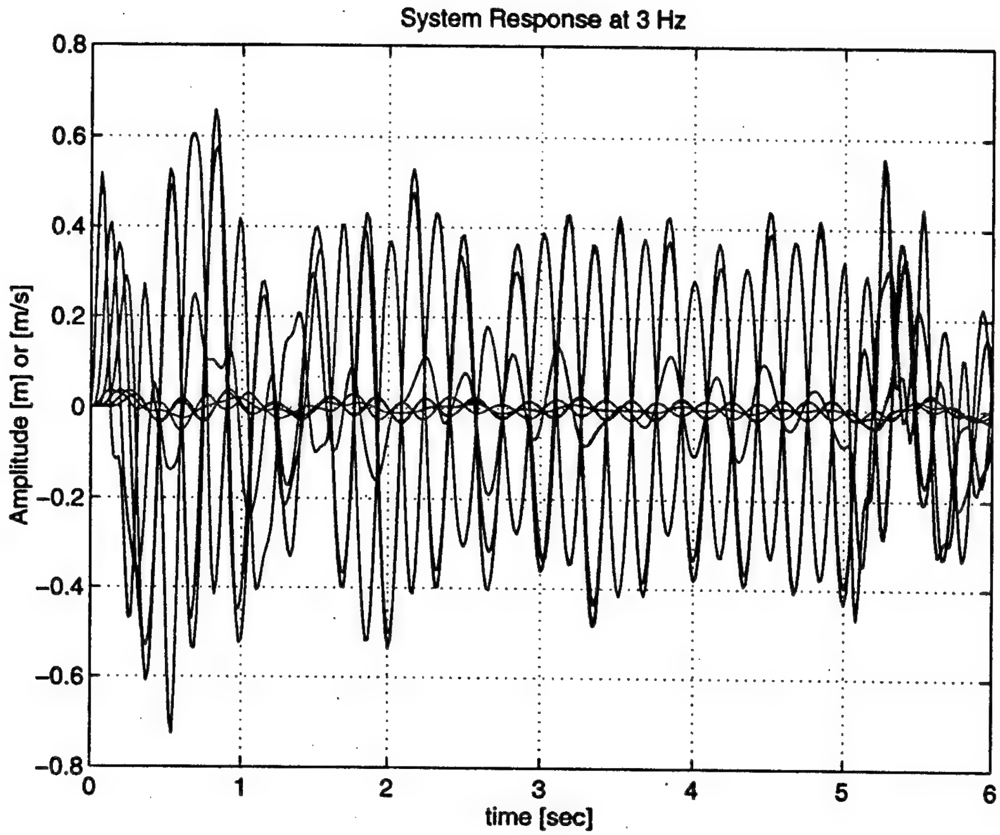


Figure 4.1: Full Response, Configuration 1

Figure 4.2 breaks the state variables down and displays them. This gives a feel for the coupling involved in the system.

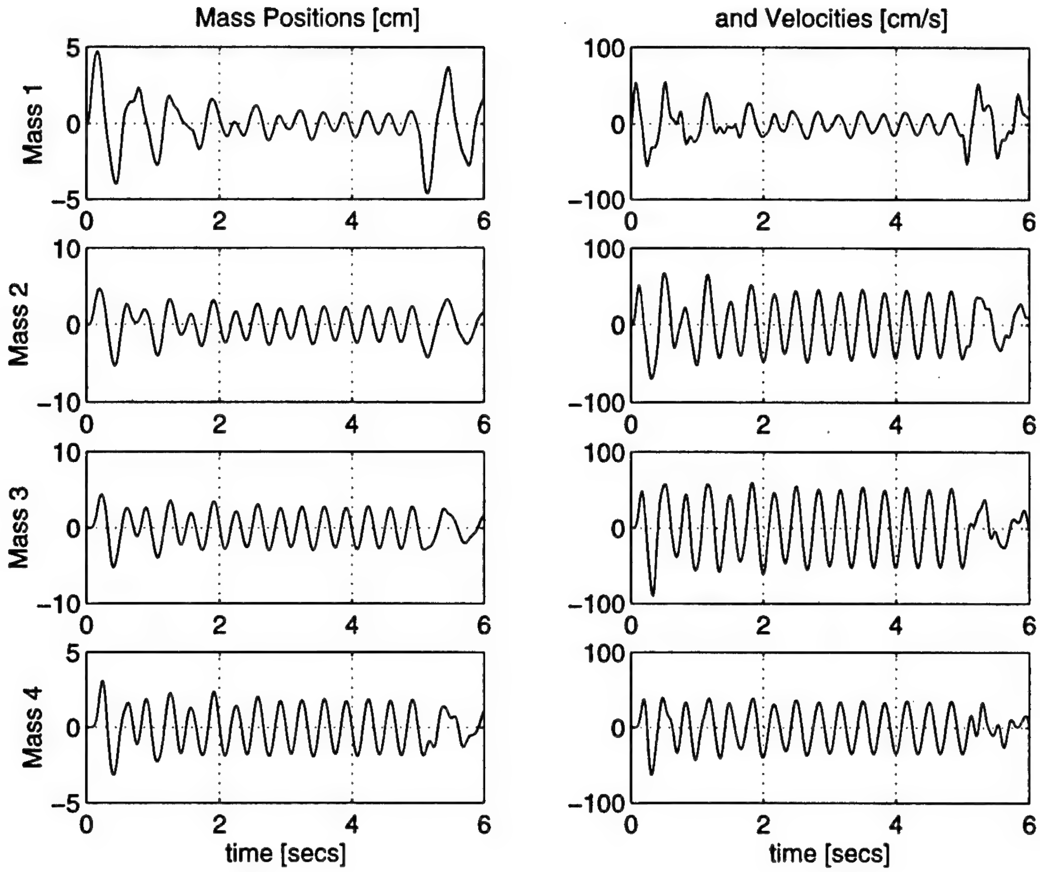


Figure 4.2: Full Response, Separated State Variables

Also of interest are the root locus plots associated for each of the state variables. These are shown in Figures 4.3 and 4.4.

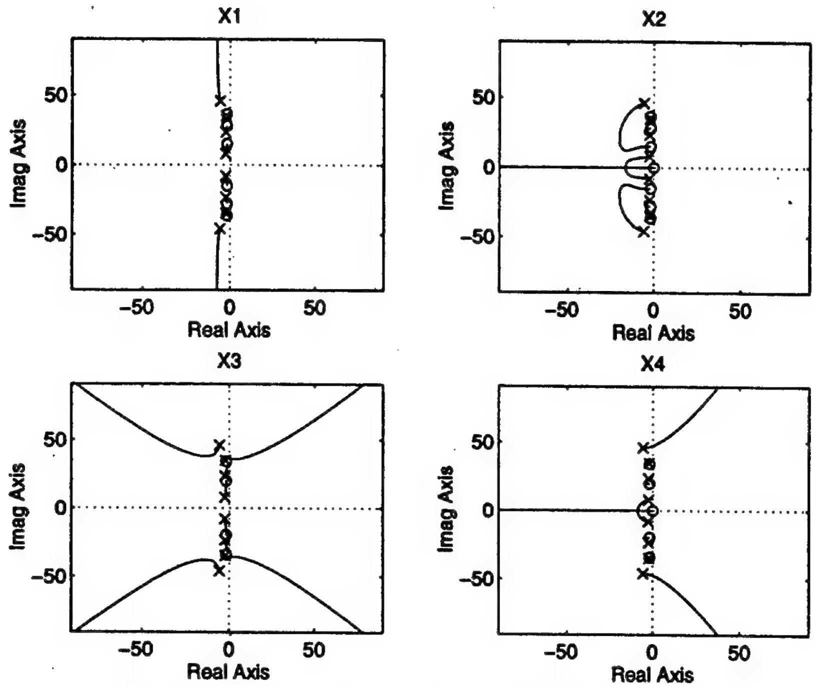


Figure 4.3: Root Locus Plots for X1 through X4

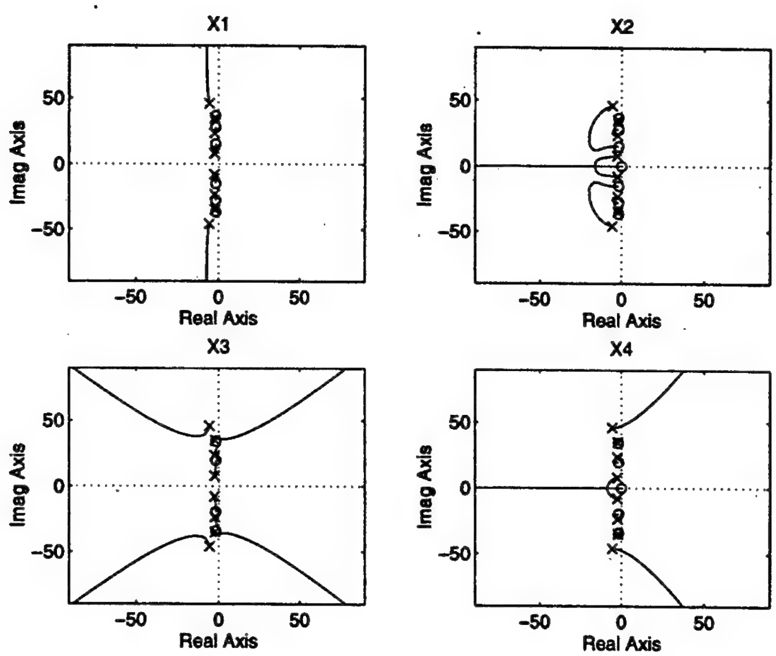


Figure 4.4: Root Locus Plots for X5 through X8

4.2 Model Comparison

In order to compare the mathematical model with the actual RECS testbed, similar data must be analyzed over a range of input frequencies. To gain an accurate comparison, a transfer function bode plot of the testbed is obtained by using a Digital Signal Analyzer (DSA). This is then compared with transfer functions of the theoretical model in Matlab. Figures 4.5 shows the DSA-derived transfer function from the input force to the position of Mass 4. This is obtained with 0.454 kg masses on sliders #2, 3, and 4. Slider #1 was left as just the slider weight of 335 g. This transfer function was obtained using a sine sweep at 450 mV. The figure shows the four modes in the system and the phase response we would expect for four poles.

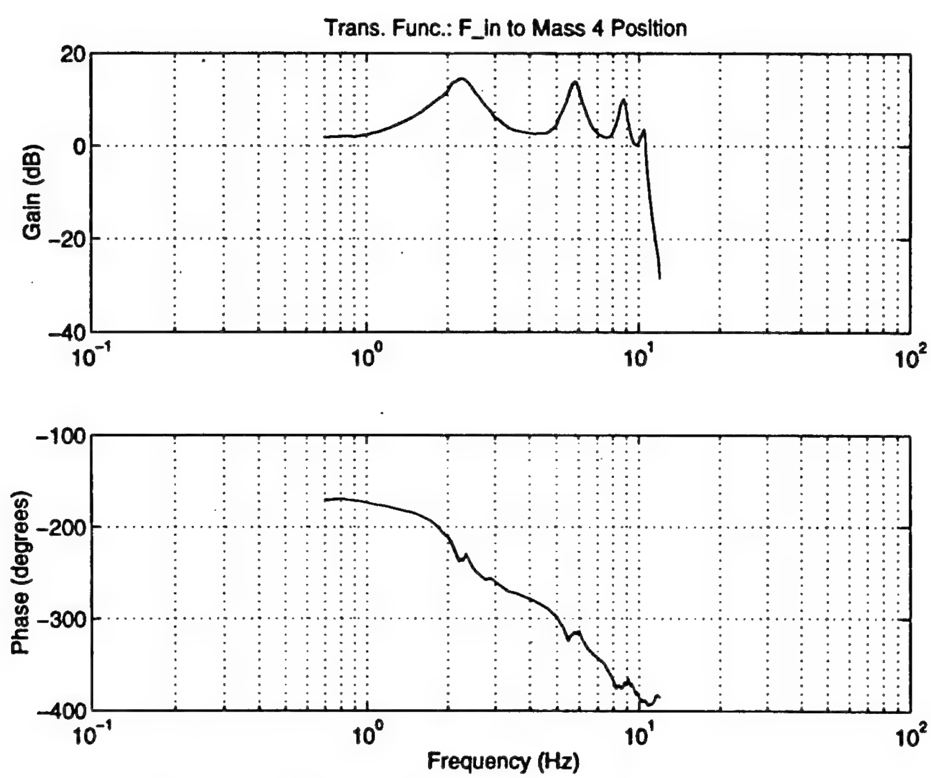


Figure 4.5: Transfer Function; RECS Testbed (DSA)

Transfer functions were taken with a DC offset and without, sweeping up and sweeping down, and with different springs. The same basic shape seen in Figure 4.3 remained throughout the tests.

To get a theoretical transfer function in Matlab, the nominal values of all masses and springs were entered into the system matrices of Chapter 3. This yielded a transfer function for an ideal system with no friction as shown in Figure 4.6 with the spring constant set to its stated value of k = 560 N/m.

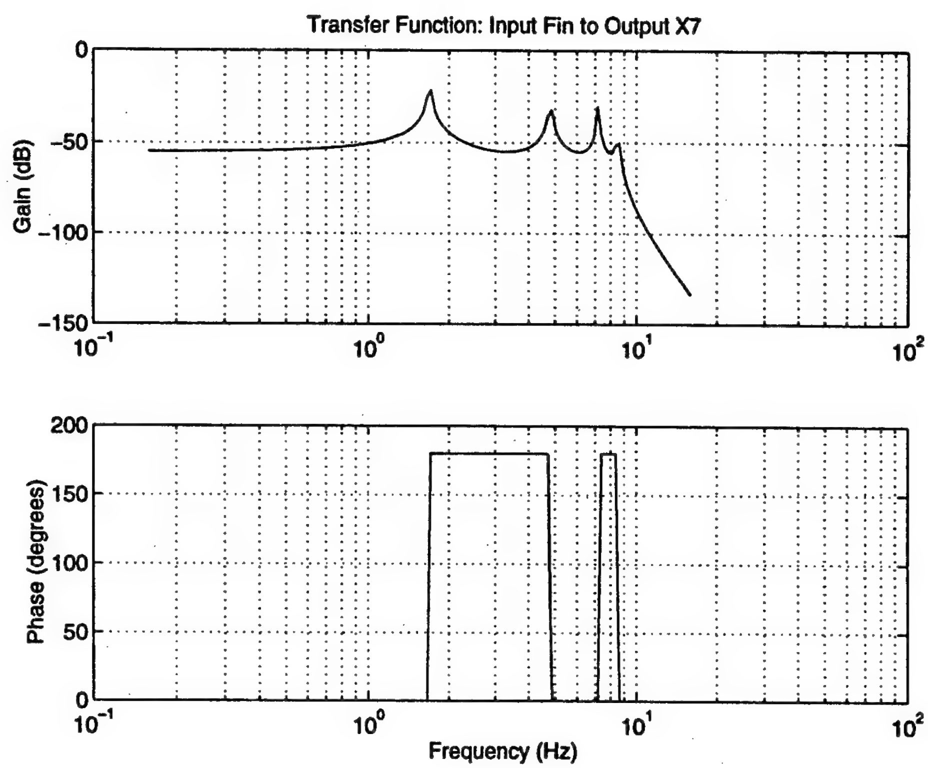


Figure 4.6: Transfer Function; Theoretical Model

The differences between them were mostly resolved through some adjustment to the theoretical model's parameters. Primarily, a set of constant friction terms were

added as discussed in Chapter 3. This new 'matched' model and the DSA results are compared in Figure 4.7.

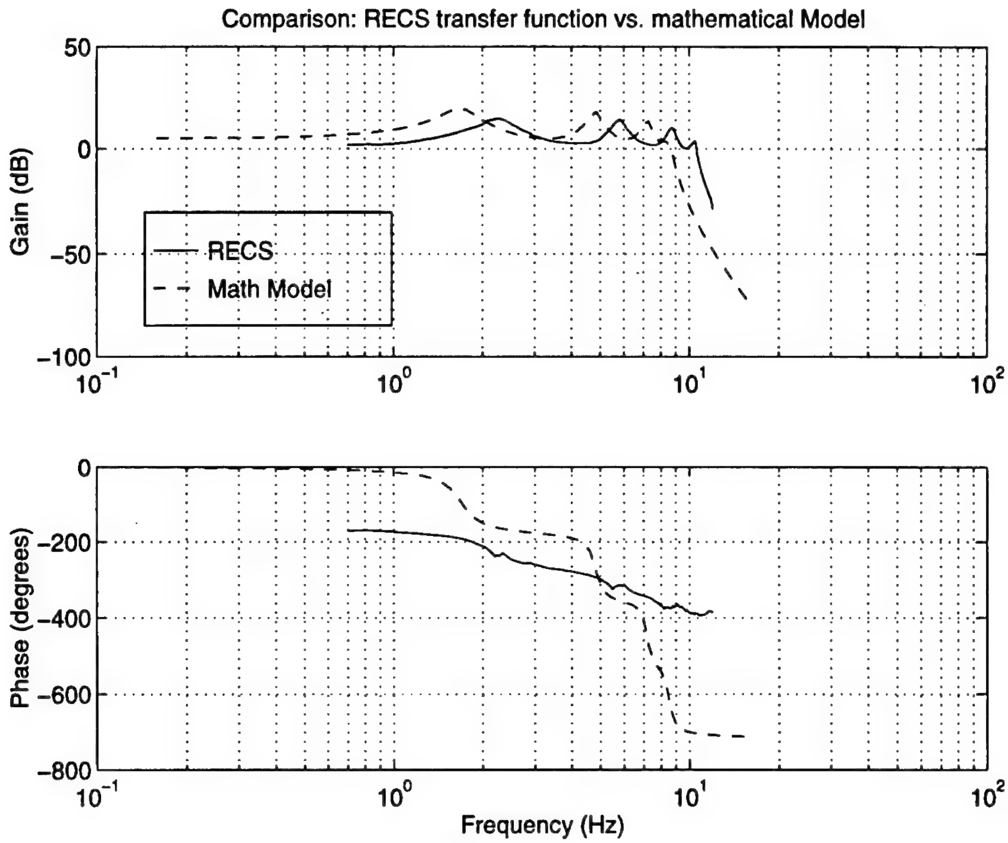


Figure 4.7: Transfer Function Comparison

As shown, this model will effectively represents the real hardware.

Since RECS can be conformed in an endless combination of parameters, many modeling development exercises are possible. For the one listed above, the system parameters are listed in Table 4.1.

Table 4.1: Parameters of Model Fit

Parameter	Measured Value	'Matched' Value
M1	0.335 Kg	‘
M2	0.789 Kg	‘
M3	0.789 Kg	‘
M4	0.789 Kg	‘
K1	560 N/m	‘
K2	560 N/m	‘
K3	560 N/m	‘
K4	560 N/m	‘
Fmu1	--	3.5 N/(m/s)
Fmu2	--	3.5 N/(m/s)
Fmu3	--	3.5 N/(m/s)
Fmu4	--	3.5 N/(m/s)

Chapter 5

CONTROLLER DEVELOPMENT

5.1 Configuration 1: System Identification

Several control block diagrams are presented here to help the prospective controller developer get started on the RECS Testbed. Figure 5.1 shows the RECS testbed with labels for all known and predicted control flow. While under most circumstances several simplifications can be made, it is good to refer back to the complete diagram for analysis. Note that the actuator group operates in closed loop. This is necessary for the servo system to work together. For normal RECS operation that group can be considered to be a single-input, single-output block and the closed loop characteristics can be ignored. This figure depicts the configuration for all plant identification simulations with the RECS Testbed.

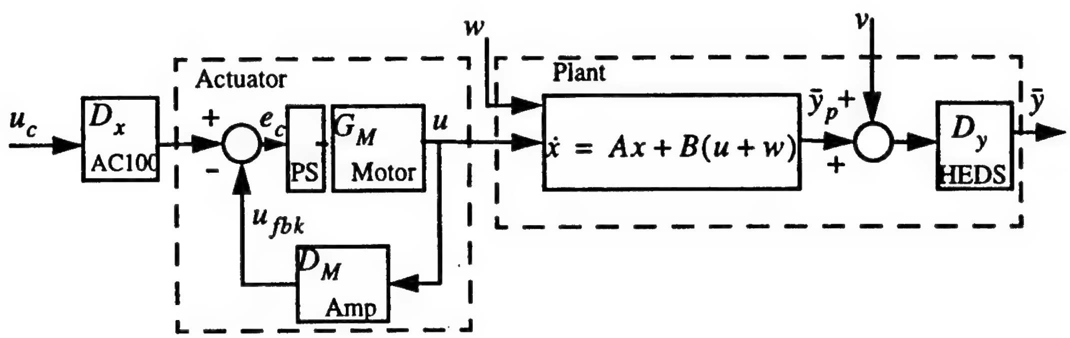


Figure 5.1: Configuration 1: Systems ID Mode

5.2 Configuration 1: Closed Loop Operation

Figures 5.2 and 5.3 show RECS in the two primary operations where the DC motor would be employed as the control actuator. The first is for simulations of regulation. In this type of testing, the drive signal is zero, but initial conditions cause the system to be displaced from the desired position.

In the second configuration, designed for tracking problems, there may or may not still be initial conditions, but the input is now nonzero. The challenge for the controller is still to make the output match the input.

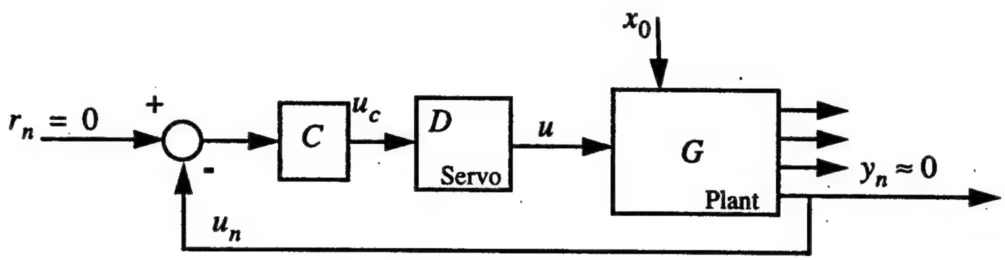


Figure 5.2: Regulation

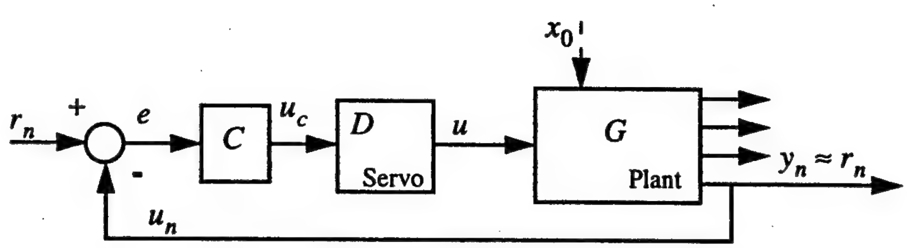


Figure 5.3: Tracking

5.3 Configuration 2: Closed Loop Operation

As described in previous chapters, the RECS testbed is designed to be used in a second configuration that yields interesting simulations for input disturbance rejection. Figure 5.4 shows the control diagram for this set up.

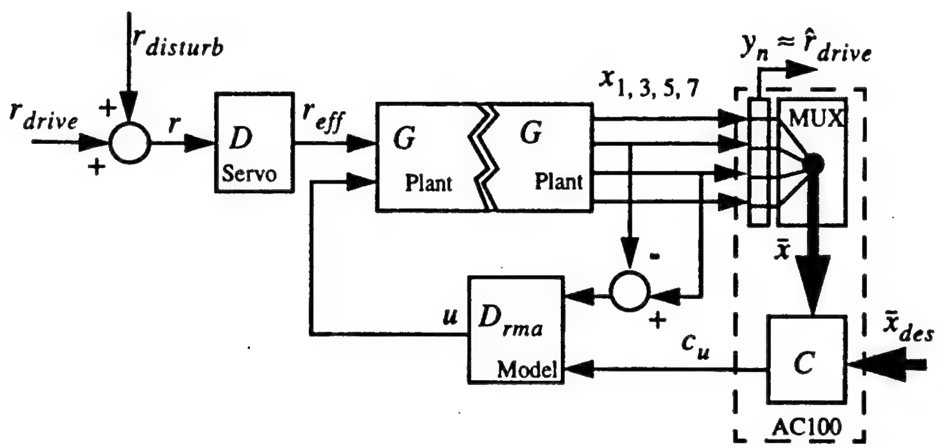


Figure 5.4: Configuration 2: Input Disturbance Rejection

The basic theory of operation for this setup is discussed in Chapter 2 and the mathematical modeling is covered in Chapter 3. The plant is shown with a broken line tó designate the fact that, although it is one coupled system, the control actuation input is inject mid-system. This is how input disturbances can be effectively rejected by the system. The sign changes resulting from the input u being inserted in the middle of plant G is accounted for in the state space matrices of Chapter 2.

Chapter 6

CONCLUSIONS AND FUTURE DISCUSSION

6.1 Conclusions

This thesis was begun with the objective of presenting a functional reconfigurable testbed to the UW CRSL to augment its present controller development hardware. Once that was completed, additional time was spent examining the mathematical models and implementation challenges. The RECS testbed takes the textbook example of a rectilinear elastically-coupled system (spring/mass/damper arrangement) and makes a real world hardware-in-the-loop platform. Combined with the Xmath software and the AC100, the RECS testbed is a fast and flexible environment for controller development and verification. As discussed in the introduction, this is valuable for confronting the nonlinearities of real world controllers.

6.2 Future Development

This thesis was intended to be the foundation of future development or academic demonstration. The following are some suggestions to maximize the value for either application.

6.2.1 GUI Interactive Program

The accompanying Matlab GUI program should be of great interest for those developing structures or control course curricula (for example, UW's current EE/AA 446 or EE/AA/ME 448). This program entitled RECSgui.m was originally developed by Clint Schneider and Kalev Sepp. It was then debugged and made operational by

Nicholas Hardman (all of UW CRSL). It can be found on the AA Department's system as

- ~hardman/project/mfiles/RECSgui.m

or it can berecreated from the included code in Appendix C.

This program is limited to the study of one mass spring and damper, but it demonstrates the value and ease of use of GUI programs. Full control of the system parameters is given by interactive slider bars. For any change in parameters the step response, impulse response, initial conditions response, and pole/zero plot are automatically updated in their respective display windows.

For help in producing a final draft version of this program there is a great deal of information on the internet from Matlab's site.

6.2.2 Future Study

As stated in the introduction, a RECS testbed was designed from the start with the desire for maximum reconfigurability. This was to be done without compromise in performance in any particular configuration. The discussion of Chapter 4 shows that this was successfully achieved, and now it is up to the user to take advantage of the potential in this testbed. As discussed in Chapter 2, the Xmath software and AC100 provide a flexible interface environment so that controller development can be performed in everything from classical to fuzzy control theory. Regardless of the format, the following bullets state some areas where the RECS testbed could be used for implementation and validation of future control systems.

- System transfer functions can be found of second, third, or fourth-order sys-

tems with either the Digital Signal Analyzer or Xmath. This will work as either a student lab problem or demonstration tool.

- Robustness tests are very easily performed since all masses and springs can be quickly modified.
- Sensitivity studies can be done with some sensor modification to accommodate sensor transfer function adjustment.
- Nonlinear parameter modeling can be performed with the use of precise parameter components. For instance, if springs and masses of precisely known values are installed, the resulting system responses can be compared to theoretical models to isolate the nonlinear components.

Once again, it must be stressed that the reconfigurability of the testbed is not limited to parameter variations. The system is wired to accept modifications to sensor outputs and force inputs in order to accommodate some unforeseen future desired format. Of course, all physical modifications require corresponding software changes, but with this in mind, the possibilities are endless.

REFERENCES

[1] Chrisman, Karl, "Coplanar Inverted Pendulum and Pivot Arm", Master's Thesis, University of Washington, 1994.

[2] De Silva, Clarence W., Control Sensors and Actuators, Prentice Hall, Inc., Englewood Cliffs, NJ, 1989.

[3] Franklin, Gene F., Powell, David J., Emami-Naeini, Abbas, Feedback Control of Dynamic Systems, 3rd ed., Addison-Wesley Publishing Co., New York, NY, 1994.

[4] Ling Dynamic Systems, "Installation and Operating Manual", 2nd ed., Amendment 9, Royston, England, Jan 1995.

[5] Saner, Floyd E., "Servo Motor Application Notes", Pittman, Harleysville, PA, 1990.

[6] Young, Hugh D., University Physics, 8th ed., Addison-Wesley Publishing Company, Inc., New York, NY, 1992.

[7] Sepp, Kalev, "Robust Control of Linear Track Cart-Pendulum", Master's Thesis, University of Washington, 1994.

[8] Leahy, Jennifer Martin, "Control of Nonlinear Underactuated Systems: The Rotary Arm Inverted Pendulum Problem", Master's Thesis, University of Washington, 1995.

[9] Pittman, "Bulletin 5000", Harleysville, PA, 1989.

Appendix A

WIRING

Tables A.1, A.2, and A.3 give the information on every pin of the RECS components. It also includes unconnected pins and possible future applications. Figure A.1 gives the information contained in Table A.1 in schematic form. As noted previously, several sections are unused at this time. These connectors have been installed, and are intended to give flexibility for future possible configurations. The control box schematic is included in Figure A.2 It has already been discussed in Chapter 2. Following the control box schematic is Table A.4 with all of the components that go into the box.

The connectors used with the Pittman Power Supply, Servo Amplifier Card, and DC motor are shown in Figure A.3. These are rather nonstandard connecting formats, so compatibility is not guaranteed unless future projects have connectors adaptable to these Pittman products. Also, shown are the pinouts for the IDC and DB-9 connectors.

Tables A.5 and A.6 are taken from the RealSim books and give further information on those pinouts.

Table A.1: Wiring List

Power Sup	ply	Servo-	Amp		Motor/ Encoder	
P.S. Function	#	Color	#	Color	#	Function
+5 VDC	1	Red-white	b4			
+12 VDC	2	Green-white	b2			
-12 VDC	3	Blue-white	d2			
GND (logic)	4	Black-white	z2			
Motor Supply	13	Orange-black	d30			
Motor GND	15	Red	z32			
Pwr Interlock	17	Green-black	b16			
Pwr (GND)	19	Blue-red	z16			
				Motor		
			d26	Brown	1	Phase A
			z24	Red	2	B
			d22	Orange	3	C
			b10	Grey	5	Hall Sensor A
			d10	Blue	6	B
			d8	White	7	C
			b8	Green	8	D (Lo-Cog opt)
			d4	Violet	9	5 VDC
			d&b 14	Black	10	GND
			DB 9	Optical Enc	oder	
5 VDC	5	Blue	2	Red	1to4/4	Vcc
GND (logic)	8	Blue-black	9	Black	1to4/1	GND
			6	Braid/Lt.Grn	--	Shield GND

Table A.2: AC100 Connections

	RMA	Servo Amp	Encoders		In/Out	IP-DAC6	IP QUAD
Function	#	#	slider/pin	Color		#	#
GND	White			Black	o	1	--
Signal *	Orng			White	o	2	--
*Note: From: ac-100 to kepco analog amp to rma							
Ctrl Signal GND		z14		Blk	o	7	--
+/-10V Signal#		d16		White	o	8	--
#Note: From ac-100 to servo-amp							
Analog GND						13	--
Sensor #1 (Analog)					o	14	--
Analog GND						19	--
Sensor #2 (Analog)					o	20	--
Enc. #1 Ch. A pin3			3	Green	i	--	3
(motor slider) Ch. B 5			8	White	i	--	7
Enc. #2 Ch. A 3			3	Green	i	--	15
Ch. B 5			8	White	i	--	19
Enc. #3 Ch. A 3			3	Green	i	--	27
Ch. B 5			8	White	i	--	31
Enc. #4 Ch. A 3			3	Green	i	--	39
Ch. B 5			8	White	i	--	43
Encoder Note: Enc.#1 is on motor slider. Pin #1 is designated by a triangle			gnd	blk			46

Table A.3: Unconnected Pins List

Unit	#	Function	Color	Function
Optical Enc.s	1to4/2	NC		
Motor	4	NC	Yellow	
		Motor Encs		
	11	Ch. A	Lt. Blue	i
	12	INV Ch. A	Lt. Violet	i
	13	Ch. B	Yellow	i
	14	INV Ch. B	Orange	i
	15	Index	Green	i
	16	INV Index	White	i
Servo-Amp	z12	I(t) Mntr	Can be a St. Var	
Power Supply	6	12 VDC	White-black	
	7	-12 VDC	Black-red	
	10	NC	Black	
	11	NC	White	
	12	NC	White-red	
	14	NC	Green	
	16	NC	Orange-red	
	18	SafeGND	Red-black	

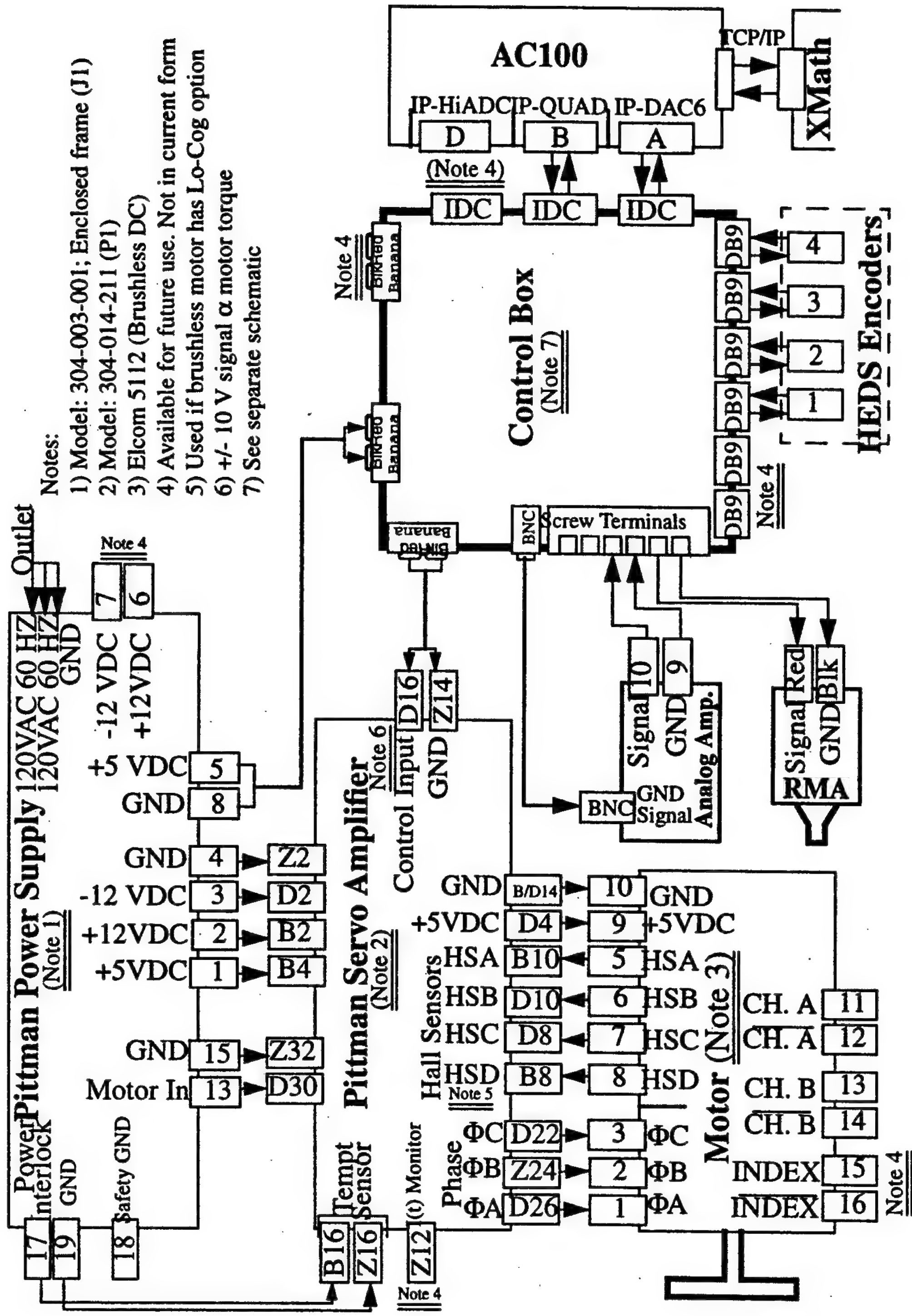


Figure A.1: RECS Schmatic

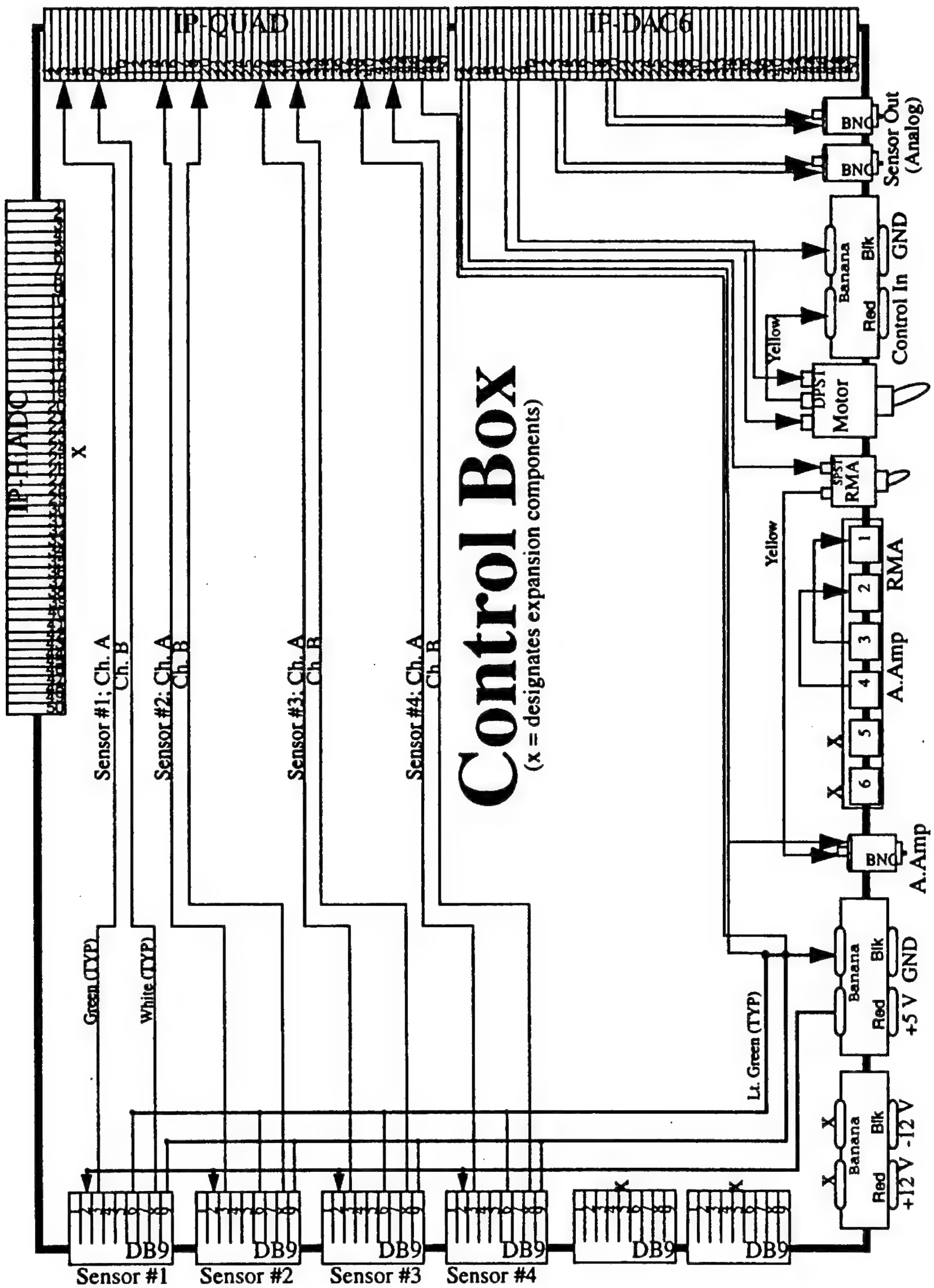
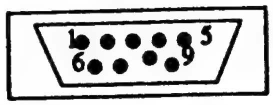


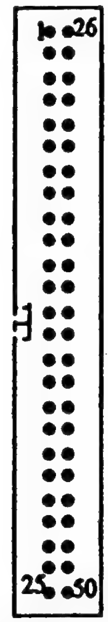
Figure A.2: Control Box Schmatic

Table A.4: Control Box Components

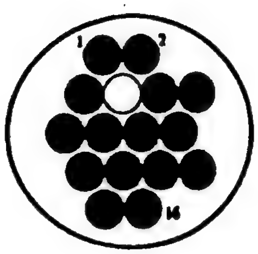
Item Description	Spec.s	Qty	Purpose
Box, Chassis w/ cover:	10"x5"x3.5"	1	Container
Alum. Mount	8-32 thread	1	Mount to testbed
Saftey switch	DTSP	1	Motor Control
Saftey switch	STSP	1	RMA Control
Connectors:			
Screw Mounts	6 post	1	signal
IDC, 50 pin	notched, female	3	AC100 cards
Coax Connector	BNC, female	1	A. Amp signal
DB-9, 9 pin	female	6	Sensors
Dual Banana	female, red/blk	3	Power, Signal



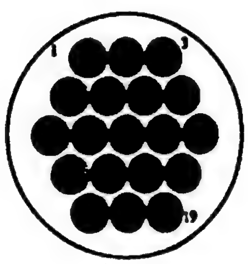
Optical Encoder Connector (DB-9)



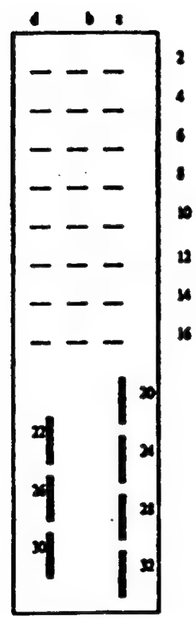
AC-100 Connectors (IDC)



Brushless Motor Connector
(Motor to Servo Amplifier)



Power Supply Connector
(Power Supply to Servo Amplifier)



Servo Amplifier Connector P1

Figure A.3: Connectors

Table A.5: AC100, IP-QUAD Connections

	Function I/O			Function I/O		
	RS-422	Logic	Pin #	RS-422	Logic	Pin #
Channel 1	X+	no connect	1	GND	GND	2
	X-	X	3	GND	GND	4
	Y+	no connect		GND	GND	6
	Y-	Y	7	GND	GND	8
	Z+	no connect	9	GND	GND	10
	Z-	Z	11	GND	GND	12
Channel 2	Y+	no connect	17	GND	GND	18
	Y-	Y	19	GND	GND	20
	Y-	Y	19	GND	GND	20
	Z+	no connect	21	GND	GND	22
	Z-	Z	23	GND	GND	24
	X+	no connect	13	GND	GND	14
	X-	X	15	GND	GND	16
Channel 3	X+	no connect	25	GND	GND	26
	X-	X	27	GND	GND	28
	Y+	no connect	29	GND	GND	30
	Y-	Y	31	GND	GND	32
	Z+	no connect	33	GND	GND	34
	Z-	Z	35	GND	GND	36
Channel 4	X+	no connect	37	GND	GND	38
	X-	X	39	GND	GND	40
	Y+	no connect	41	GND	GND	42
	Y-	Y	43	GND	GND	44
	Z+	no connect	45	GND	GND	46
	Z-	Z	47	GND	GND	48
	8 MHz +		49	8 MHz	GND	50

Table A.6: AC100, IP-DAC6 Connections

Function	I/O	Function	I/O
Analog Ground DAC 1 Output	1 2	DAC 5 Sense Analog Ground	27 28
DAC 1 Sense DAC 1 Current In	3 4	n/c n/c	29 30
n/c DAC 1 Current Out	5 6	Analog Ground DAC 6 Output	31 32
Analog Ground DAC 2 Output	7 8	DAC 6 Sense Analog Ground	33 34
DAC 2 Sense DAC 2 Current In	9 10	n/c n/c	35 36
n/c DAC 2 Current Out	11 12	Analog Ground n/c	37 38
Analog Ground DAC 3 Output	13 14	n/c Analog Ground	39 40
DAC 3 Sense Analog Ground	15 16	n/c n/c	41 42
n/c n/c	17 18	Analog Ground n/c	43 44
Analog Ground DAC 4 Output	19 20	n/c Analog Ground	45 46
DAC 4 Sense Analog Ground	21 22	n/c Analog Ground	47 48
n/c n/c	23 24	+5.000 V Drive (Input/Output)	49
Analog Ground DAC 5 Output	25 26	n/c	50

Appendix B

AC100/Matrix$_x$ IMPLIMENTATION

B.1 RECS Testbed Checklist

A general operating checklist is given on the following four pages. **The most recent version of this document is to be left with the testbed and should be strictly adhered to for safety reasons.** The importance of some instructions is not immediately obvious. Deviations and permanent alterations to this checklist should only be done after consulting Chapter 2 and Appendix A for the component specifications and warnings.

B.2 AC100 Layout

Following the checklist are several figures. Figure B.1 shows the GUI window for the Interactive Animation of the AC100 interface. This is a basic layout that gives the user input alteration and output monitoring abilities. The slider bars are adjustable with the mouse buttons. The plots continuously update and provide visual references for system behavior. This interactive window is highly modifiable and can be accessed by opening the `RECS.pic` file in the Matrix$_x$ Animation Builder. The remaining figures are of the SystemBuild project: RECS which is located on capri.aa under the `~/nick_hardman/RECS` directory. This is a graphical representation of the code that runs behind the scenes to make the interface work.

RECS TESTBED OPERATING CHECKLIST

Nicholas Hardman, DEC 97

RECS testbed Setup

1 Insure all slider mounts are secured to platform and sliders move free.

2 Tighten all springs and mass mounts for secure and safe operation.

3 (Optional) Fasten RMA signal wires so as not to obstruct slider.

4 Set motor gear teeth and insure motor is plugged in.

5 Check all control box connections and switches

- 5a Connect IP-QUAD and IP-DAC6 ribbon cables to the correct IDC connectors. (labeled on the control box)
- 5b Turn motor and RMA switches off.

  Turn on individually before activating with external inputs (Primarily intended for safety shutoff switches).
- 5c The analog sensor bnc connections will be unused unless DSA is connected.
- 5d Connect analog amp. to both the input and output on the box.
- 5e Connect 5v power and motor control signal from servo-amp.

6 Check servo-amp and power supply connections.

7 Check other external evaluation equipment.

8 Turn Master power switch on (on the power strip).

Xmath Setup

1 Run Xmath software on X-terminal that will interface with the AC100.

(Currently “Eagle” or “Condor”)

- 1a Type ‘`xhost capri`’ in login shell.
- 1b Type ‘`telnet capri.aa`’
- 1c Type ‘`ac100`’; and password
- 1d Type ‘`setenv DISPLAY eagle:0`’ (or condor)
- 1e If desired, ‘`olwm&`’ for Sun OS X-window manager
- 1f Type ‘`tcsh`’ for T-shell, com line completion
- 1g Change directory to ‘`nick_hardman/RECS`’
- 1h Type ‘`rsim50`’

2 Other possibilities:

2a If changes are needed in the c_c30.hce file:

 2a.1 Original (on PC): c:/ac100dsp/c30_sp/c_c30.hce

 2a.2 Edit files on capri using: `textedit`

AC100 Setup

1 Turn on PC hosting the AC100. At DOS prompt enter '`ac100svr`'.

For Xmath Controller Development

1 Select [`Xmath/SystemBuild`]

2 Load a file containing block diagrams: '`load "recs.xmd"`'

3 [Build/Catalog] ; move to a particular superblock
or [Build/ Edit superblock]

4 Mouse commands

 4a To open a superblock -- 2Xclick on block ID # (upper rt.)

 4b To flip block horizontally -- 2xclick on body

 4c To go to upper level of superblock -- right click on Superblock ID bar (white bar with block info.)

5 Select [`Build/Gen. Real-Time Code`]; for the DSP
[RECS]; select from list, creates RECS.rtf and RECS.c

6 Type '`save "RECS.xmd"`'; if update is desired for blocks, variables
and/or '`quit`'; To close SystemBuild

7 RealSim GUI shows buttons to press:

 7a [Autocode] (Be sure to first type '`ac100svr`' on PC)

 7b [Compile and Link]

 7c [Animation Builder]

 7d [Load Pict];Accept

 7e [RTF Names]

 7f [Labels ON]

 7g [Save Pict]

8 Other modules:

 8a Hardware Connection Editor -- Alter block inputs and outputs

 8b Interactive Animation Display -- Adjust scaling factors

Xmath Controller Execution

1 Type '`rsim50`'. The GUI will appear.
2 [Download & Run];to run experiment, initialize
3 [Start Controller]; run RealSim controller, verify
4 Set the user inputs:
   4a Drive signal, Disturbance signal
   4b Sensor setup
5 Select `[Start Controller]`
6 To exchange info. with Matlab:
   6a Use `xmload.m` to convert from Matrixx to Matlab
   6b `savems.m` transfers from Matlab to Matrixx format.

Digital Signal Analyzer (DSA) and AC100 Setup

This is an optional section, but may be desirable for quick and simple examination of the system by bypassing the need for the Xmath software. This can be done using a sine sweep, random noise analysis, or fft on the system.

1 For a transfer function of the system, enter the keys shown below.

Table 1: Obtain Transfer Function

Hard key	1st Soft key	2d Soft Key
preset	do preset	2 channel
meas data	freq. resp.	
disp. format	bode diag.	
scale	autoscale on	
active trace	autoscale on	
input	chan.1 setup	in. low gnd
input	chan.2 setup	in. low gnd
source	rand. noise	(or sine swp)
	level	.2 Vpk
freq'	span	100 Hz

Table 1: Obtain Transfer Function

Hard key	1st Soft key	2d Soft Key
avg	average on	
	# averages	20, enter
source	on	

2 Transfer data for use in Matlab.

2a Save file on 3.5", 1.4 Mb, DOS floppy as a *.dat

Table 2: Save data

Hard key	1st Soft key	2d Soft Key
save/recall	save data	
	save trace	
	into file	
	clear entry	*filename.dat*
active trace		

Stores data on screen in two vectors:

vec1 -complex number (both mag and phase)

vec1x -frequency in hertz.

2b Transport to PC that has '*SDFTOML*'.

2c Open DOS window. Copy *.dat files to c:/DSA

2d In DSA directory, run '`sdftoml filename.dat /x /b`' which creates *.mat files.

2e FTP to UNIX account and plot in Matlab.

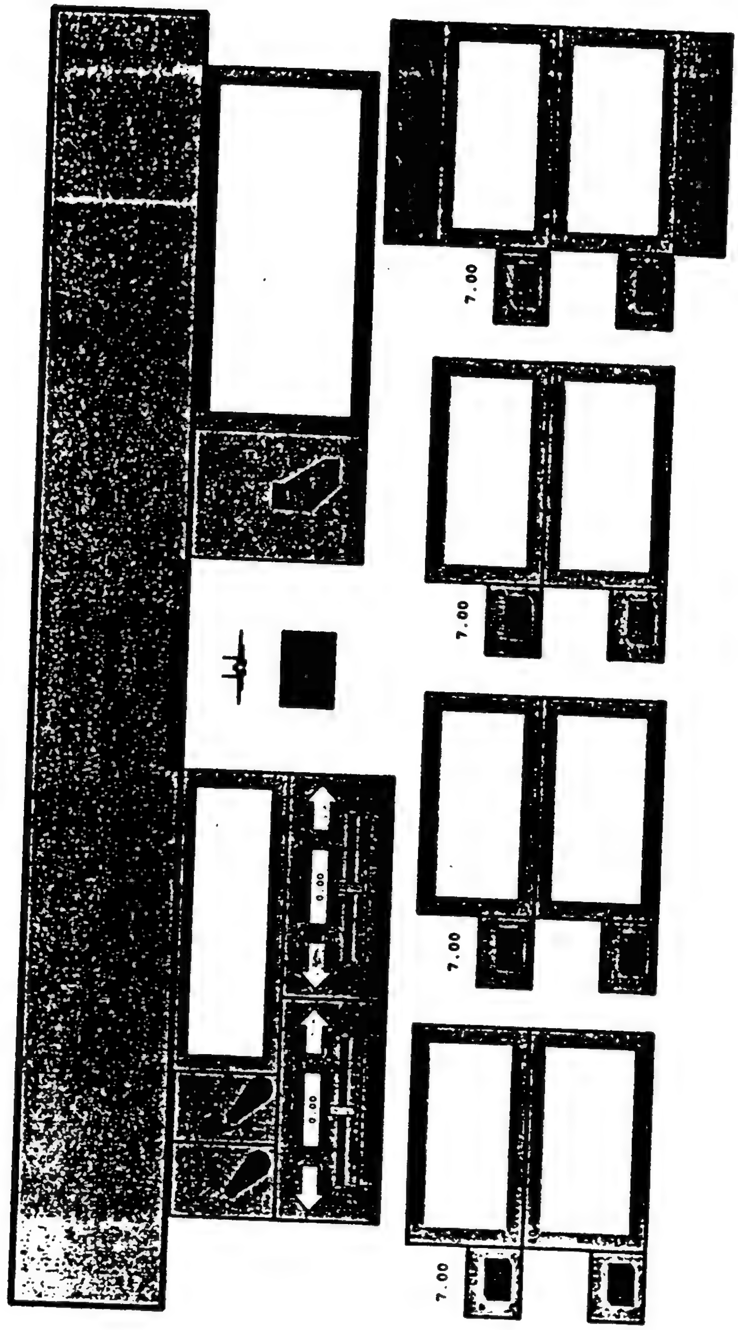


Figure B.1: Interactive Animation Window

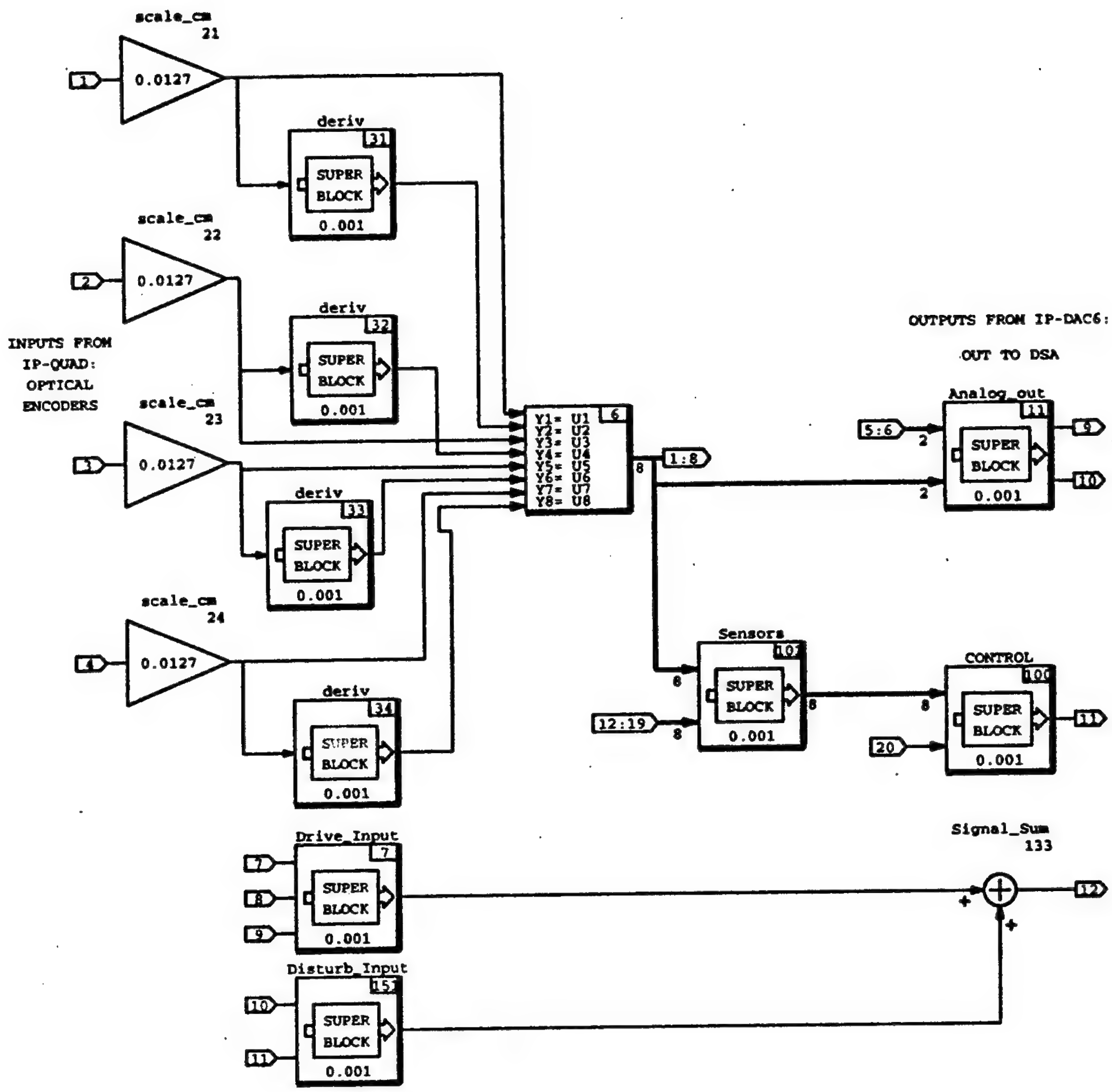


Figure B.2: RECS Superblock

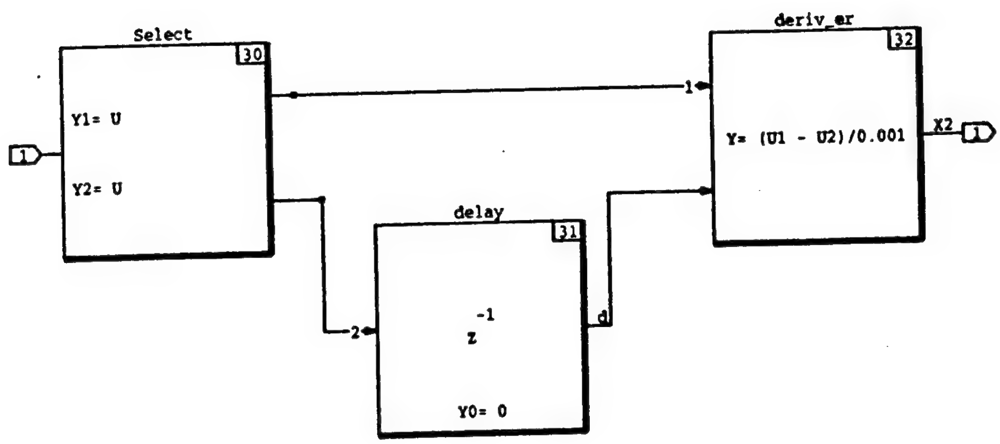


Figure B.3: Derivative Superblock

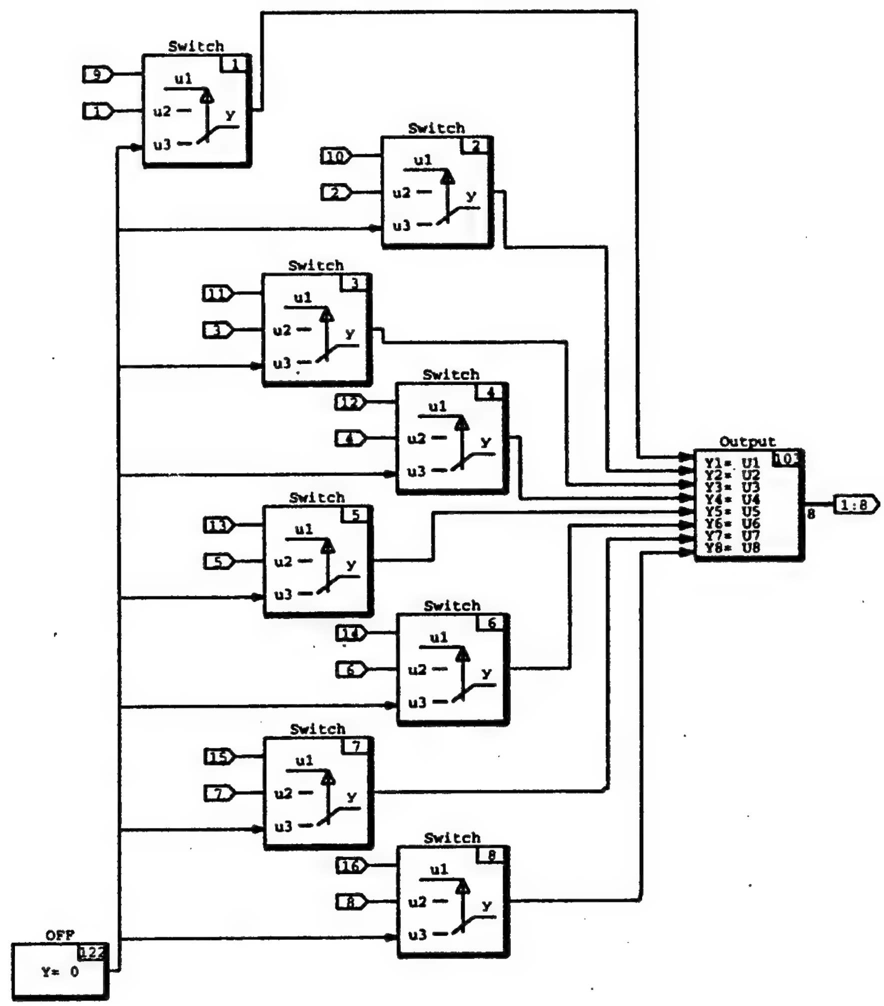


Figure B.4: Sensors Superblock

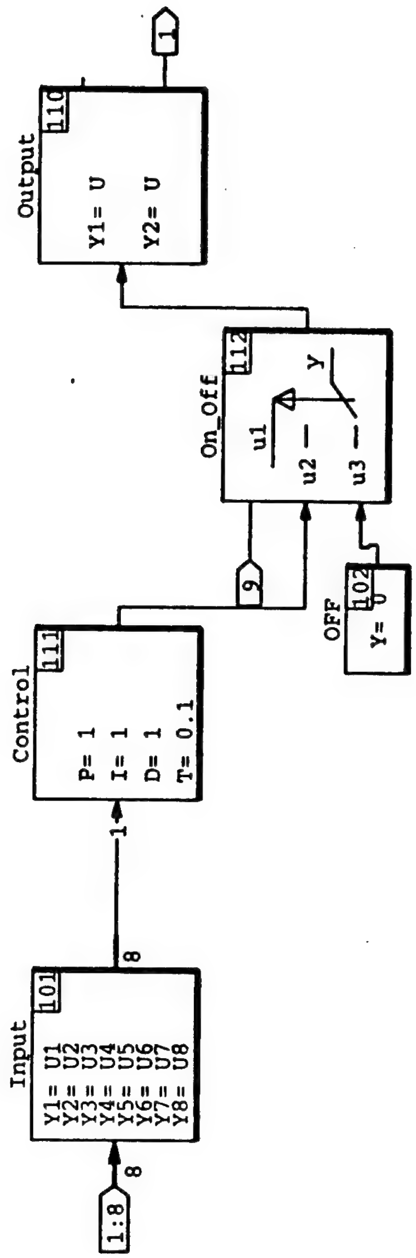


Figure B.5: Control Superblock

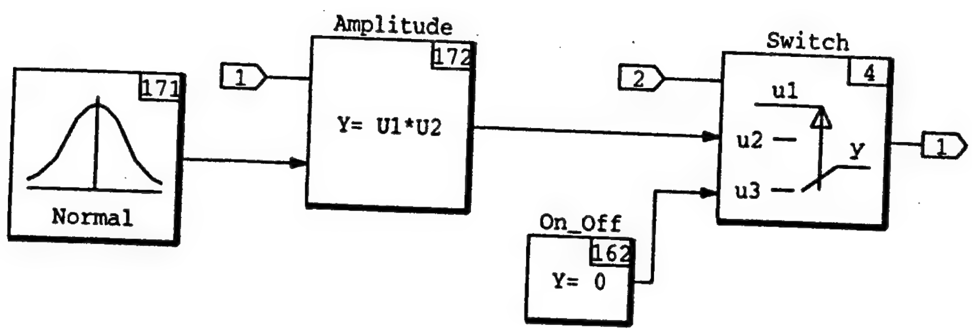


Figure B.6: Disturb_Input Superblock

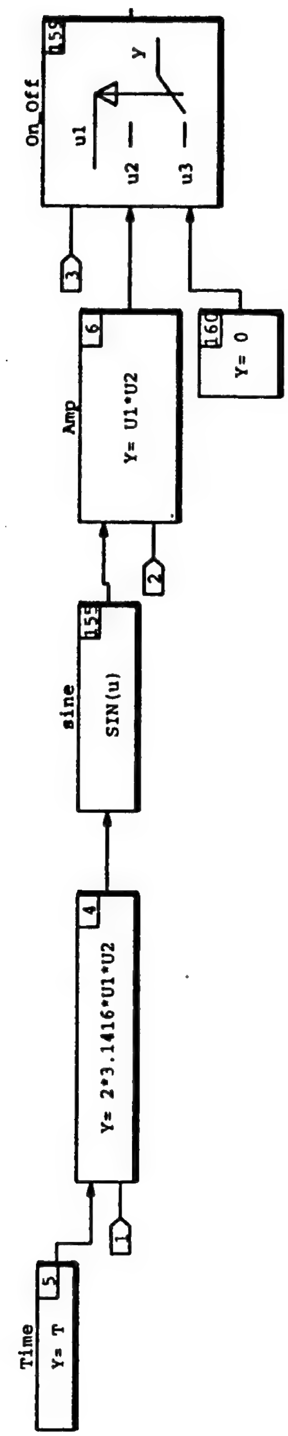


Figure B.7: Drive_Input Superblock

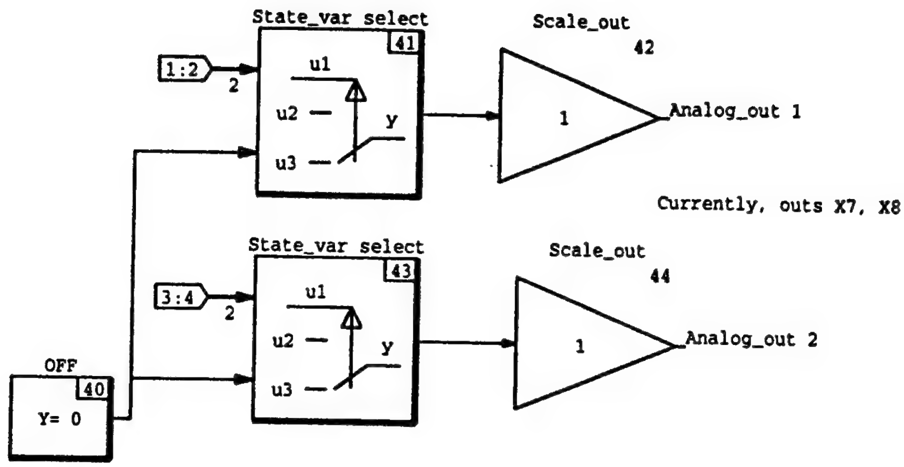


Figure B.8: Analog_Out Superblock

Appendix C

Matlab Simulation Program Files

C.1 Model Simulation File Index

The following pages give the Matlab script files for the simulations presented in Chapter 4. These were run in Matlab 4.0. The first five programs were written by Nicholas Hardman. The original drafts of the last three programs were written by Clint Schneider at the University of Washington. After modification, they are now useful as theoretical analogs to the RECS testbed. The Matlab GUI simulation discussed in Chapter 6 is one of these programs.

Table C.1: Associated Matlab files

#	File Name:	Function:
1	master.m	Master RECS file; Prints transfer functions of simulations and DSA data
2 3	bode_dsa.m bodeplus.m	Function files for putting data in bode plot form
4	config1.m	Creates the plant model for configuration 1
5	config2.m	Creates the plant model for configuration 2
6	c1sim.m	Simulink file for configuration 1
7	c2sim.m	Simulink file for configuration 2
8	RECSgui.m	Matlab GUI for Spring/Mass/Damper System

C.2 File: Master.m

```
% Name: ~/project/mfiles/master.m
% Rectilinear Vibration Experiment Master M-file
%The following mfiles are associated with this project and comprise the entire
%   package of files.
% master.m   ----> (this file) plots the DSA data and compares it with the
%                  simulation data.  This file works for both
%                  config1 and config2
%
%
% config1.m ----> Creates the continuous plant model for configuration 1.
%                  and all of the plots
%                  (RMA inactive. Springs between all masses).
%
% c1sim.m   ----> Simulink file for configuration 1.
%
%
% config2.m ----> Creates the continuous plant model for configuration 2.
%                  (RMA active, rod  between masses 2 & 3).
%
% c2sim.m   ----> Simulink file for configuration 2.
%by Nicholas Hardman
%


% setup
% don't clear all;
close all;

%input('Choose configuration to run: (1 or 2)?','config');
config = 1;   %for now 'cuz this part's not working

do1=1;if do1==1;
% DSA plots of transfer function:  X4/Fin----------------------------

      %if config == 1;
          bode_dsa(1,'tf_dsa4.mat');
          subplot(211);
          title('Trans. Func.: F_in to Mass 4 Position');
          %bode_dsa(2,'tf_vel_1.mat');
          %subplot(211);
          %title('Trans. Func.: F_in to Mass 4 Velocity');
      %elseif config == 2;
      %   bode_dsa(1,'tf_pos_2.mat');
      %   subplot(211);
      %   title('Trans. Func.: F_in to Mass 4 Position');
```

```
        %   bode_dsa(2,'tf_vel_2.mat');
        %   subplot(211);
        %   title('Trans. Func.: F_in to Mass 4 Velocity');
        %else
        %   disp('Enter a 1 or a 2');
        %end%if
%ranging
%figure(1)
%subplot(211)
%axis([100 3.2e3*2*pi -50 10]);
%subplot(212)
%axis([100 3.2e3*2*pi -200 200]);

%figure(2)
%subplot(211)
%axis([100 3.2e3*2*pi -50 10]);
%subplot(212)
%axis([100 3.2e3*2*pi -200 200]);

end;     %DSA part
% Simulation plots of transfer function X4/Fin ----------------------------
do2=1;if do2==1;

        if config == 1;
            config1;%runs the matlab file to enter param.s
        elseif config == 2;
            config2;%runs the matlab file to enter param.s
        else
            disp('Enter a 1 or a 2');
        end;%if

c_p4=c(7,:); d_1=0;
[m,p,w] = bode(a,b,c_p4,d_1,1);
bodeplus(2,m*10000,p,w);
subplot(211);
title('Transfer Function: Input Fin to Output X7');

%c_v4=c(8,:);
%[m,p,w] = bode(a,b,c_v4,d_1,1);
%bodeplus(4,m,p,w);
%subplot(211);
%title('Transfer Function: Input Fin to Output X8');


end;     %model part
% Combined----------------------------------------------------------
do3=0;if do3==1;

figure(1);
subplot(211); hold on;
```

```
subplot(212); hold on;
c_p4=c(7,:);
[m,p,w] = bode(a,b,c_p4,d_1,1);
bodeplus(1,m,p,w);
subplot(211);
title('Comparison: RECS transfer function vs. mathematical Model');
grid;
%legend('-',RECS,'--','Math Model');

subplot(212)
grid;

%figure(2); hold on;
%c_v4=c(:,8);
%[m,p,w] = bode(a,b,c_v4,d,1);
%bodeplus(4,m,p,w);
%subplot(211);
%title('Comparison: RECS transfer function vs. Simulation');
%legend('+',RECS,'-.','Matlab');

end;      %combo part
```

C.3 File: bode_dsa.m

```
function bode_dsa(fig_no, filename)
%bode_dsa  plots .mat files in bode plot form
%

load (filename)
o2i1x=o2i1x(1:390,:);
o2i1=o2i1(1:390,:);
%w = 2*pi*o2i1x;
w = o2i1x;

figure(fig_no)
subplot(211)
semilogx(w, 20*log10(abs(o2i1)));
grid; %xlabel('Frequency (Hz)');
ylabel('Gain (dB)')

subplot(212)
semilogx(w, unwrap(angle(o2i1)*180/pi));
grid; xlabel('Frequency (Hz)');
ylabel('Phase (degrees)')
```

C.4 File: bodeplus.m

```
function bodeplus(fig_no,mag,phase,w)
```

```
figure(fig_no)

%use this block if inputing as a single t.f.
%mag=abs(H);
%phase=angle(H);

%w = 2*pi*w;%if given Hz and rad/sec desired
w = w/(2*pi);%If Hz desired and given rad/sec

subplot(211)
semilogx(w, 20*log10(mag));
grid
ylabel('Gain (dB)')

subplot(212)
semilogx(w, phase)
grid
xlabel('Frequency (Hz)');

ylabel('Phase (degrees)')
```

C.5 File: config1.m

```
%       4 Mass Vibration System
%
%       States: x1 = Position Mass 1
%               x2 = Velocity Mass 1
%            _  x3 = Position Mass 2
%            x = x4 = Velocity Mass 2
%               x5 = Position Mass 3
%               x6 = Velocity Mass 3
%               x7 = Position Mass 4
%               x8 = Velocity Mass 4
%
%       Parameters:k1,k2,k3,k4 Respective spring constants
%               m1,m2,m3,m4Respective masses
%               c1      Dash Pot constant
%               Fin(t)  Driving force: Pittman brushless motor
%
%Compensating factorsFd1,Fd2,Fd3,Fd4Friction factors of sliders
%               Fg      Gear meshing, power loss
%


%Define constants----------------------------------------

k1 =    560.4; %N/m
k2 =    560.4; %N/m
```

```
k3 =    560.4; %N/m
k4 =    560.4; %N/m


m1 =    0.335; %kilograms
m2 =    0.335+0.4536; %kilograms ,slider + 1 lb. wt.
m3 =    0.335+0.4536; %kilograms
m4 =    0.335+0.4536; %kilograms

c1 = 0; %N/m/s

Fd1 = 1.5; %N/(m/s)
Fd2 = 1.5; %N/(m/s)
Fd3 = 1.5; %N/(m/s)
Fd4 = 1.5; %N/(m/s)


%Driving Force parameters  (Fin)---------------
A=      10;  %N Amplitude
f=      10;  %Hz Freq.
res=    20;  %# samples per period
Trun=   9;   %secs length of signal time
Tadd=   1;   %secslength of 0 added to signal
p=      0;   %phase angle


%system Building-----------------------------------
%Define Matrices----------------------
a21 = -k1/m1;
a22 = -Fd1/m1;
a23 =  k1/m1;
a41 =  k1/m2;
a43 = -(k1+k2)/m2;
a44 = -Fd2/m2;
a45 = k2/m2;
a63 = k2/m3;
a65 = -(k2+k3)/m3;
a66 = -Fd3/m3;
a67 = k3/m3;
a85 = k3/m4;
a87 = -(k3+k4)/m4;
a88 = -(c1 + Fd4)/m4;


a = [   0  1  0  0  0  0  0  0 ;
        a21  a22  a23  0  0  0  0  0 ;
        0  0  0  1  0  0  0  0 ;
        a41  0  a43  a44  a45  0  0  0 ;
        0  0  0  0  0  1  0  0 ;
```

```
        0  0  a63 0   a65 a66 a67 0  ;
        0  0  0   0   00  0   1  ;
        0  0  0   0   a85 0   a87 a88];

b21 = 1/m1;

b = [ 0 ;
    b21 ;
     0 ;
     0 ;
     0 ;
     0 ;
     0 ;
     0 ];

c = eye(8);
d = zeros(8,1);
[N,D]=ss2tf(a,b,c,d);

%Driving Force:  Fin------------------------


T=        1/f; %sec period
fs=       f*res;%Hzsampling freq.
Ts=       1/fs;%secssample period (spacing)
cycles=  5;
n1 = [0:Ts:cycles];
F1 = A*sin((2*pi*f)*n1+p);
Fin = [F1 zeros(1,(Tadd*fs))];
n=[0:Ts:((size(Fin,2)-1)/fs)];
%plot(n,Fin);
%end drive force creation-----------------


doplot=0;if doplot==1;
figure(1);

lsim(a,b,c,d,Fin,n);
%legend('x1','x1 vel.','x2','x2 vel.','x3','x3 %vel.','x4','x4 vel.');
grid;
title('System Response at 3 Hz');
ylabel('Amplitude [m] or [m/s]'); xlabel('time [sec]');
end %doplot,plot block--------


[Y,X]=lsim(a,b,c,d,Fin,n);

doplot=0; if doplot==1;
for m=1:8;
        figure(2);
```

```
        subplot(4,2,m);
        plot(n,(X(:,m)*100));
        grid;

end;

subplot(421);
title('     Mass Positions [cm]');
ylabel('Mass 1');
subplot(422);
title('and Velocities [cm/s]');
%ylabel('Mass 1');

subplot(423);
ylabel('Mass 2');
subplot(424);
%ylabel('Mass 2');

subplot(425);
ylabel('Mass 3');
subplot(426);
%ylabel('Mass 3');

subplot(427);
ylabel('Mass 4');
xlabel('time [secs]');
subplot(428);
%ylabel('Mass 4');
xlabel('time [secs]');

end; %plot block #2


doplot=0; if doplot==1; %----------------------

figure(3)
for m=1:4;
        subplot(2,2,m);
        rlocus(a,b,c(m,:),0)
end;

subplot(221);
title('X1')
subplot(222);
title('X2')
subplot(223);
title('X3')
subplot(224);
title('X4')
```

```
figure(4)
for m=5:8;
        subplot(2,2,(m-4));
        rlocus(a,b,c(m,:),0)
end;
subplot(221);
title('X5')
subplot(222);
title('X6')
subplot(223);
title('X7')
subplot(224);
title('X8')


end; %plot block #3
```

C.6 File: config2.m

```
%       4 Mass Vibration System w/ RMA
%
%       States: x1 = Position Mass 1
%               x2 = Velocity Mass 1
%           _   x3 = Position Mass 2
%           x  = x4 = Velocity Mass 2
%               x5 = Position Mass 3
%               x6 = Velocity Mass 3
%               x7 = Position Mass 4
%               x8 = Velocity Mass 4
%
%       Parameters:k1,k3,k4 Respective spring constants
%               krma=kr RMA spring element
%               m1,m2,m3,m4Respective masses
%               c1      Dash Pot constant
%               crma=cr RMA damping element
%               Fin(t)  Driving force: Pittman brushless motor
%               Frma    Conrol force: RMA
%
%Compensating factorsFd1,Fd2,Fd3,Fd4Friction factors of sliders
%               Fg      Gear meshing, power loss
%


%Define constants--------------------------------------

k1 =    300; %N/m
kr =    300; %N/m
k3 =    300; %N/m
```

```
k4 =    300; %N/m

m1 =    1.500; %kilograms
m2 =    1.500; %kilograms
m3 =    1.500; %kilograms
m4 =    1.500; %kilograms

c1 = 0;  %N/(m/s)
cr = 4.97;%N/(m/s)
Fd1 = 1; %N/(m/s)
Fd2 = 1; %N/(m/s)
Fd3 = 1; %N/(m/s)
Fd4 = 1; %N/(m/s)


%Driving Force parameters  (Fin)----------------
A=      10;  %N Amplitude
f=      2;   %Hz Freq.
res=    20;  %#  samples per period
Trun=   9;   %secs length of signal time
Tadd=   1;   %secslength of 0 added to signal
p=      0;   %phase angle


%system Building------------------------------------
%Define Matrices-----------------------
a21 = -k1/m1;
a22 = -Fd1/m1;
a23 =  k1/m1;
a41 =  k1/m2;
a43 = -(k1+kr)/m2;
a44 = -(cr+Fd2)/m2;
a45 = kr/m2;
a46 = cr/m2;
a63 = kr/m3;
a64 = -cr/m3;
a65 = -(kr+k3)/m3;
a66 = -(cr+Fd3)/m3;
a67 = k3/m3;
a85 = k3/m4;
a87 = -(k3+k4)/m4;
a88 = -(c1 + Fd4)/m4;


a = [   0  1  0  0  0  0  0  0 ;
        a21 a22 a23 0  0  0  0  0 ;
        0  0  0  1  0  0  0  0 ;
        a41 0  a43 a44 a45 a46 0  0 ;
        0  0  0  0  0  1  0  0 ;
```

```
          0  0  a63 a64 a65 a66 a67 0  ;
          0  0  0  0  00  0  1  ;
          0  0  0  0  a85 0  a87 a88];
%
b21 = 1/m1;
b42 = -1/m2;
b62 = 1/m3;

b = [ 0 0  ;
    b21 0  ;
     0 0  ;
     0 b42 ;
     0 0  ;
     0 b62 ;
     0 0  ;
     0 0 ];

c = eye(8);
d = zeros(8,2);

[N,D]=ss2tf(a,b,c,d,1);

%Driving Force: Fin------------------------


T=      1/f; %sec period
fs=     f*res;%Hzsampling freq.
Ts=     1/fs;%secssample period (spacing)
n1 = [0:Ts:cycles];
F1 = A*sin((2*pi*f)*n1+p);
Fin = [F1 zeros(1,(Tadd*fs))];
n=[0:Ts:((size(Fin,2)-1)/fs)];
%plot(n,Fin);
%end drive force creation----------------


%Total Force: Ttot-----------------------
Frma = zeros(size(Fin));

Ftot = [Fin' Frma'];


doplot=1;if doplot==1;
figure(1);
legend('x1','x1 vel.','x2','x2 vel.','x3','x3 vel.','x4','x4 vel.');
lsim(a,b,c,d,Ftot,n);
grid;
title('System Response');
ylabel('Amplitude [cm] or [cm/s]'); xlabel('time [sec]');
end %doplot,plot block--------
```

```
[Y,X]=lsim(a,b,c,d,Ftot,n);

doplot=1; if doplot==1;
for m=1:8;
        figure(2);
        subplot(4,2,m);
        plot(n,X(:,m));
        grid;

end;

subplot(421);
title('     Configuration 2: Mass Positions [cm]');
ylabel('Mass 1');
subplot(422);
title('and Velocities [cm/s]');
%ylabel('Mass 1');

subplot(423);
ylabel('Mass 2');
subplot(424);
%ylabel('Mass 2');

subplot(425);
ylabel('Mass 3');
subplot(426);
%ylabel('Mass 3');

subplot(427);
ylabel('Mass 4');
xlabel('time [secs]');
subplot(428);
%ylabel('Mass 4');
xlabel('time [secs]');

end; %plot block #2
```

C.7 File: c1sim.m

```
function [ret,x0,str,ts,xts]=config1sim(t,x,u,flag);
%CONFIG1SIMis the M-file description of the SIMULINK system named CONFIG1SIM.
%       The block-diagram can be displayed by typing: CONFIG1SIM.
%
%       SYS=CONFIG1SIM(T,X,U,FLAG) returns depending on FLAG certain
%       system values given time point, T, current state vector, X,
%       and input vector, U.
%       FLAG is used to indicate the type of output to be returned in SYS.
%
```

```
%       Setting FLAG=1 causes CONFIG1SIM to return state derivatives, FLAG=2
%       discrete states, FLAG=3 system outputs and FLAG=4 next sample
%       time. For more information and other options see SFUNC.
%
%       Calling CONFIG1SIM with a FLAG of zero:
%       [SIZES]=CONFIG1SIM([],[],[],0), returns a vector, SIZES, which
%       contains the sizes of the state vector and other parameters.
%           SIZES(1) number of states
%           SIZES(2) number of discrete states
%           SIZES(3) number of outputs
%           SIZES(4) number of inputs
%           SIZES(5) number of roots (currently unsupported)
%           SIZES(6) direct feedthrough flag
%           SIZES(7) number of sample times
%
%       For the definition of other parameters in SIZES, see SFUNC.
%       See also, TRIM, LINMOD, LINSIM, EULER, RK23, RK45, ADAMS, GEAR.

% Note: This M-file is only used for saving graphical information;
%     after the model is loaded into memory an internal model
%     representation is used.

% the system will take on the name of this mfile:
sys = mfilename;
new_system(sys)
simver(1.3)
if (0 == (nargin + nargout))
   set_param(sys,'Location',[13,143,1052,766])
   open_system(sys)
end;
set_param(sys,'algorithm',     'Linear')
set_param(sys,'Start time',    '0.0')
set_param(sys,'Stop time',     '5')
set_param(sys,'Min step size', '0.01')
set_param(sys,'Max step size', '0.01')
set_param(sys,'Relative error','1e-3')
set_param(sys,'Return vars',   '')

add_block('built-in/Clock',[sys,'/','Clock'])
set_param([sys,'/','Clock'],...
              'position',[350,560,370,580])

add_block('built-in/To Workspace',[sys,'/','To Workspace1'])
set_param([sys,'/','To Workspace1'],...
              'mat-name','time',...
              'buffer','10000',...
              'position',[465,562,515,578])

add_block('built-in/State-Space',[sys,'/','State-Space'])
set_param([sys,'/','State-Space'],...
```

```
                'A','a',...
                'B','b',...
                'C','c',...
                'D','d',...
                'X0','[-0.02 0 0 0 0 0]',...
                'position',[430,234,505,276])

add_block('built-in/Note',[sys,'/','F1 (motor)'])
set_param([sys,'/','F1 (motor)'],...
                'position',[190,210,195,215])

add_block('built-in/Sine Wave',[sys,'/','Sine Wave'])
set_param([sys,'/','Sine Wave'],...
                'position',[95,165,115,185])

add_block('built-in/To Workspace',[sys,'/','To Workspace2'])
set_param([sys,'/','To Workspace2'],...
                'mat-name','f1',...
                'buffer','10000',...
                'position',[395,192,445,208])

add_block('built-in/To Workspace',[sys,'/','To Workspace'])
set_param([sys,'/','To Workspace'],...
                'mat-name','states',...
                'buffer','10000',...
                'position',[580,247,630,263])

add_block('built-in/Step Fcn',[sys,'/','Step Input'])
set_param([sys,'/','Step Input'],...
                'Time','.1',...
                'After','0',...
                'position',[185,245,205,265])

add_block('built-in/Signal Generator',[sys,'/',['Signal',13,'Generator']])
set_param([sys,'/',['Signal',13,'Generator']],...
                'Peak','1.000000',...
                'Peak Range','5.000000',...
                'Freq','1.000000',...
                'Freq Range','5.000000',...
                'Wave','Sin',...
                'Units','Rads',...
                'position',[80,258,125,292])
add_line(sys,[375,570;460,570])
add_line(sys,[510,255;575,255])
add_line(sys,[210,255;275,255;275,200;390,200])
add_line(sys,[275,255;415,255;425,255])

drawnow

% Return any arguments.
```

```
if (nargin | nargout)
        % Must use feval here to access system in memory
        if (nargin > 3)
            if (flag == 0)
                eval(['[ret,x0,str,ts,xts]=',sys,'(t,x,u,flag);'])
            else
                eval(['ret =', sys,'(t,x,u,flag);'])
            end
        else
            [ret,x0,str,ts,xts] = feval(sys);
        end
else
        drawnow % Flash up the model and execute load callback
end
```

C.8 File: c2sim.m

```
function [ret,x0,str,ts,xts]=c2sim(t,x,u,flag);
%C2SIMis the M-file description of the SIMULINK system named C2SIM.
%       The block-diagram can be displayed by typing: C2SIM.
%
%       SYS=C2SIM(T,X,U,FLAG) returns depending on FLAG certain
%       system values given time point, T, current state vector, X,
%       and input vector, U.
%       FLAG is used to indicate the type of output to be returned in SYS.
%
%       Setting FLAG=1 causes C2SIM to return state derivatives, FLAG=2
%       discrete states, FLAG=3 system outputs and FLAG=4 next sample
%       time. For more information and other options see SFUNC.
%
%       Calling C2SIM with a FLAG of zero:
%       [SIZES]=C2SIM([],[],[],0), returns a vector, SIZES, which
%       contains the sizes of the state vector and other parameters.
%           SIZES(1) number of states
%           SIZES(2) number of discrete states
%           SIZES(3) number of outputs
%           SIZES(4) number of inputs
%           SIZES(5) number of roots (currently unsupported)
%           SIZES(6) direct feedthrough flag
%           SIZES(7) number of sample times
%
%       For the definition of other parameters in SIZES, see SFUNC.
%       See also, TRIM, LINMOD, LINSIM, EULER, RK23, RK45, ADAMS, GEAR.

% Note: This M-file is only used for saving graphical information;
%     after the model is loaded into memory an internal model
%     representation is used.

% the system will take on the name of this mfile:
```

```
sys = mfilename;
new_system(sys)
simver(1.3)
if (0 == (nargin + nargout))
   set_param(sys,'Location',[491,442,1530,1065])
   open_system(sys)
end;
set_param(sys,'algorithm',    'Linear')
set_param(sys,'Start time',   '0.0')
set_param(sys,'Stop time',    '5')
set_param(sys,'Min step size', '0.01')
set_param(sys,'Max step size', '0.01')
set_param(sys,'Relative error','1e-3')
set_param(sys,'Return vars',  '')

add_block('built-in/State-Space',[sys,'/','State-Space'])
set_param([sys,'/','State-Space'],...
              'A','a',...
              'B','b',...
              'C','c',...
              'D','d',...
              'X0','[-0.02 0 0 0 0 0]',...
              'position',[430,234,505,276])

add_block('built-in/Mux',[sys,'/','Mux'])
set_param([sys,'/','Mux'],...
              'inputs','2',...
              'position',[325,236,355,269])

add_block('built-in/Constant',[sys,'/','Constant'])
set_param([sys,'/','Constant'],...
              'Value','0',...
              'position',[255,250,275,270])

add_block('built-in/Note',[sys,'/','F1 (motor)'])
set_param([sys,'/','F1 (motor)'],...
              'position',[190,210,195,215])

add_block('built-in/Note',[sys,'/','F2 (RMA)'])
set_param([sys,'/','F2 (RMA)'],...
              'position',[265,285,270,290])

add_block('built-in/Sine Wave',[sys,'/','Sine Wave'])
set_param([sys,'/','Sine Wave'],...
              'position',[95,165,115,185])

add_block('built-in/Step Fcn',[sys,'/','Step Input'])
set_param([sys,'/','Step Input'],...
              'Time','.1',...
              'After','0',...
```

```
                'position',[185,235,205,255])

add_block('built-in/To Workspace',[sys,'/','To Workspace2'])
set_param([sys,'/','To Workspace2'],...
                'mat-name','f1',...
                'buffer','10000',...
                'position',[395,192,445,208])

add_block('built-in/Derivative',[sys,'/','Derivative'])
set_param([sys,'/','Derivative'],...
                'position',[260,50,290,70])

add_block('built-in/To Workspace',[sys,'/','To Workspace'])
set_param([sys,'/','To Workspace'],...
                'mat-name','states',...
                'buffer','10000',...
                'position',[580,247,630,263])

add_block('built-in/Signal Generator',[sys,'/',['Signal',13,'Generator']])
set_param([sys,'/',['Signal',13,'Generator']],...
                'Peak','1.000000',...
                'Peak Range','5.000000',...
                'Freq','1.000000',...
                'Freq Range','5.000000',...
                'Wave','Sin',...
                'Units','Rads',...
                'position',[100,228,145,262])

add_block('built-in/To Workspace',[sys,'/','To Workspace1'])
set_param([sys,'/','To Workspace1'],...
                'mat-name','time',...
                'buffer','10000',...
                'position',[310,342,360,358])

add_block('built-in/Clock',[sys,'/','Clock'])
set_param([sys,'/','Clock'],...
                'position',[195,340,215,360])
add_line(sys,[220,350;305,350])
add_line(sys,[510,255;575,255])
add_line(sys,[360,255;425,255])
add_line(sys,[280,260;320,260])
add_line(sys,[210,245;320,245])
add_line(sys,[275,245;275,200;390,200])

drawnow

% Return any arguments.
if (nargin | nargout)
        % Must use feval here to access system in memory
        if (nargin > 3)
```

```
            if (flag == 0)
                eval(['[ret,x0,str,ts,xts]=',sys,'(t,x,u,flag);'])
            else
                eval(['ret =', sys,'(t,x,u,flag);'])
            end
        else
            [ret,x0,str,ts,xts] = feval(sys);
        end
else
        drawnow % Flash up the model and execute load callback
end
```

C.9 File: RECSgui.m

```
function RECSgui(action)
%
%RECSgui.m a simulation for the study of
%Rectilinear Elastically-Coupled Systems (Spring Mass Damper)
%
% Original version: Clint Schneider & Kalev Sepp  May 20, 1996
%         (Originally named: smd3.m)
% Debugged and Modified by: Nicholas Hardman  NOV 31, 1997
%Work done while graduate students in
%the University of Washington Control and Robotic Systems Lab
%         UW CRSL
%
%
% actions
% 'start'
% 'gainedit'
% 'gainslider'
% 'model'
% 'paramupdate'
% 'redraw'

global a b c d x0
global K M C F wn Z
global MASTER fig PZMAP STEP_RESP IMPL_RESP IC_RESP
global STEP_PLOT IMPL_PLOT IC_PLOT ROOT_PLOT
global massedit nfreqtext zedatext ...
          dampedit springedit dampslide massslide ...
          springslide Nfreqtext Dratiotext masstext ...
          damptext springtext closeme closeout

if nargin<1,
  action = 'start';
end

if strcmp(action,'start')
```

```
% graphics initialization

% Plant description
K=500;
M=1.1;
C=10;
wn=sqrt(K/M);
Z=C/2*sqrt(1/(K*M));
x0=[-.02 0];

%figure name
        fig = figure('position',[ 11 850 1251 168 ],...
             'resize','on',...
             'tag','userinterface',...
             'Name','Spring/Mass/Dashpot System Dynamics Control Panel',...
             'visible','on');

% MASTER = figure('Name','Spring/Mass/Dashpot System Dynamics Interface');
set(gcf,'Units','normalized','backingstore','off');

%Create Spring Constant blocks

        closeme = uicontrol(...
         'CallBack','RECSgui(''closeout'')',...
         'Position',[ 0.862 0.553 0.066 0.201 ],...
         'String','EXIT',...
         'Value',[0],...
         'Style','pushbutton',...
         'Units','normalized',...
         'Tag','closeme',...
         'UserData','');


        massedit = uicontrol(...
             'CallBack','RECSgui(''gainedit'')',...
             'Position',[ 0.11 0.5 0.08 0.14 ],...
             'Style','edit',...
             'String',num2str(M),...
             'Units','normalized',...
             'Tag','massedit',...
             'UserData','');
        dampedit = uicontrol(...
             'CallBack','RECSgui(''gainedit'')',...
             'Position',[ 0.372 0.504 0.078 0.146 ],...
             'Style','edit',...
             'String',num2str(C),...
             'Units','normalized',...
             'Tag','dampedit',...
             'UserData','');
        springedit = uicontrol(...
```

```
        'CallBack','RECSgui(''gainedit'')',...
        'Position',[ 0.24 0.504 0.078 0.146 ],...
        'Style','edit',...
        'String',num2str(K),...
        'Units','normalized',...
        'Tag','springedit',...
        'UserData','');
  dampslide = uicontrol(...
        'CallBack','RECSgui(''gainslider'')',...
        'Position',[ 0.36 0.296 0.102 0.114 ],...
        'Style','slider',...
        'Units','normalized',...
        'Value',[ 10 ],...
        'Min',0,...
        'Max',500,...
        'Tag','dampslide',...
        'UserData','');
  massslide = uicontrol(...
        'CallBack','RECSgui(''gainslider'')',...
        'Position',[ 0.096 0.291 0.102 0.114 ],...
        'Style','slider',...
        'Units','normalized',...
        'Value',[ 1.1 ],...
        'Min',0.001,...
        'Max',25,...
        'Tag','massslide',...
        'UserData','');
  springslide = uicontrol(...
        'CallBack','RECSgui(''gainslider'')',...
        'Position',[ 0.23 0.296 0.102 0.114 ],...
        'Style','slider',...
        'Units','normalized',...
        'Value',[ 0.2 ],...
        'Min',0.001,...
        'Max',1500,...
        'Tag','springslide',...
        'UserData','');
Nfreqtext = uicontrol(...
        'CallBack','',...
        'Position',[ 0.673 0.736 0.12 0.16 ],...
        'String','Natural Frequency (wn)',...
        'Style','text',...
        'Units','normalized',...
        'Tag','Nfreqtext',...
        'UserData','');
Dratiotext = uicontrol(...
        'CallBack','',...
        'Position',[ 0.543 0.736 0.12 0.16 ],...
        'String','Damping Ratio',...
        'Style','text',...
```

```
            'Units','normalized',...
            'Tag','Dratiotext',...
            'UserData','');
    masstext = uicontrol(...
            'CallBack','',...
            'Position',[ 0.09 0.742 0.119 0.16 ],...
            'String','Mass (M)',...
            'Style','text',...
            'Units','normalized',...
            'Tag','masstext',...
            'UserData','');
    damptext = uicontrol(...
            'CallBack','',...
            'Position',[ 0.35 0.742 0.12 0.16 ],...
            'String','Damping Constant (C)',...
            'Style','text',...
            'Units','normalized',...
            'Tag','damptext',...
            'UserData','');
    springtext = uicontrol(...
            'CallBack','',...
            'Position',[ 0.22 0.737 0.12 0.17 ],...
            'String','Spring Constant (K)',...
            'Style','text',...
            'Units','normalized',...
            'Tag','springtext',...
            'UserData','');
    nfreqtext = uicontrol(...
            'CallBack','',...
            'Position',[ 0.697 0.505 0.078 0.146 ],...
            'String',num2str(wn),...
            'Style','text',...
            'Units','normalized',...
            'Tag','nfreqtext',...
            'UserData','');
    zedatext = uicontrol(...
            'CallBack','',...
            'Position',[ 0.567 0.499 0.078 0.146 ],...
            'String',num2str(Z),...
            'Style','text',...
            'Units','normalized',...
            'Tag','zedatext',...
            'UserData','');


PZMAP = figure('Name','System Pole Zero Locations',...
            'Position',[11 420 600 355]);
IC_RESP = figure('Name','System Response to Initial Conditions X0',...
            'Position',[630 420 630 355]);
STEP_RESP = figure('Name','System Response to Step Input',...
```

```
              'Position',[11 11 600 365]);
IMPL_RESP = figure('Name','System Response to an Impulse',...
              'Position',[630 11 630 365]);

 RECSgui('model')
 RECSgui('redraw')

elseif strcmp(action,'model')
 a=[0 1;-K/M -C/M];
 b=[0;1];
 c=eye(2);
 d=[0;0];


elseif strcmp(action,'gainslider')
 K = get(springslide,'Value');
 set(springedit,'String',num2str(K));
 C = get(dampslide,'Value');
 set(dampedit,'String',num2str(C));
 M = get(massslide,'Value');
 set(massedit,'String',num2str(M));


 RECSgui('paramupdate')
 RECSgui('model')
 RECSgui('redraw')

elseif strcmp(action,'closeout')
 clear
 close all

elseif strcmp(action,'gainedit')
 K = str2num(get(springedit,'String'));
 set(springslide,'Value',K);
 C = str2num(get(dampedit,'String'));
 set(dampslide,'Value',C);
 M = str2num(get(massedit,'String'));
 set(massslide,'Value',M);

 RECSgui('paramupdate')
 RECSgui('model')
 RECSgui('redraw')

elseif strcmp(action,'paramupdate')
 wn=sqrt(K/M);
 set(nfreqtext,'String',num2str(wn));
 Z=C/2*sqrt(1/(K*M));
 set(zedatext,'String',num2str(Z));
```

```
elseif strcmp(action,'redraw')

figure(PZMAP)

pzmap(a,b,c,d)
        hold on
        v=axis;
        tempx=max(abs(v(2)),abs(v(1)));
        plot(-tempx:tempx,0*(-tempx:tempx))
        tempy=max(abs(v(4)),abs(v(3)));
        plot(0*(-tempy:tempy),(-tempy:tempy))
        clear tempx
        clear tempy
        clear v
        hold off
        grid

figure(IC_RESP)

[y,x,t]=initial(a,b,c,d,x0);
        subplot(211)
            plot(t,y(:,1),t,zeros(size(t)),'--')
            xlabel('Time (sec)')
            ylabel('Mass Position (m)');grid
        subplot(212)
            plot(t,y(:,2),t,zeros(size(t)),'--')
            xlabel('Time (sec)')
            ylabel('Mass Velocity (m/s)');grid

figure(STEP_RESP)

[y,x,t]=step(a,b,c,d,1);

        subplot(211)
            plot(t,x(:,1),'y')
            grid
            xlabel('Time (sec)')
            ylabel('Mass Position (m)')
        subplot(212)
            plot(t,x(:,2),'m')
            grid
            hold on
            v=axis;
            v2=0:v(2)/100:v(2);
            plot(v2,zeros(size(v2)),'.y')
            hold off
            xlabel('Time (sec)')
            ylabel('Mass Velocity (m/s)')
            clear v
```

```
figure(IMPL_RESP)

[y,x,t]=impulse(a,b,c,d,1);

        subplot(211)
            plot(t,x(:,1),'y')
            grid
            xlabel('Time (sec)')
            ylabel('Mass Position (m)')

        subplot(212)
            plot(t,x(:,2),'m')
            grid
            hold on
            v=axis;
            v2=0:v(2)/100:v(2);
            plot(v2,zeros(size(v2)),'.y')
            hold off
            clear v
            xlabel('Time (sec)')
            ylabel('Mass Velocity (m/s)');grid


end
end
```

Appendix D

Machined Parts Blueprints

The following figures show blueprints for some of the RECS parts that were machined at the University of Washington. These components, along with commercially sold elements, can modify or replace all of the testbed hardware.

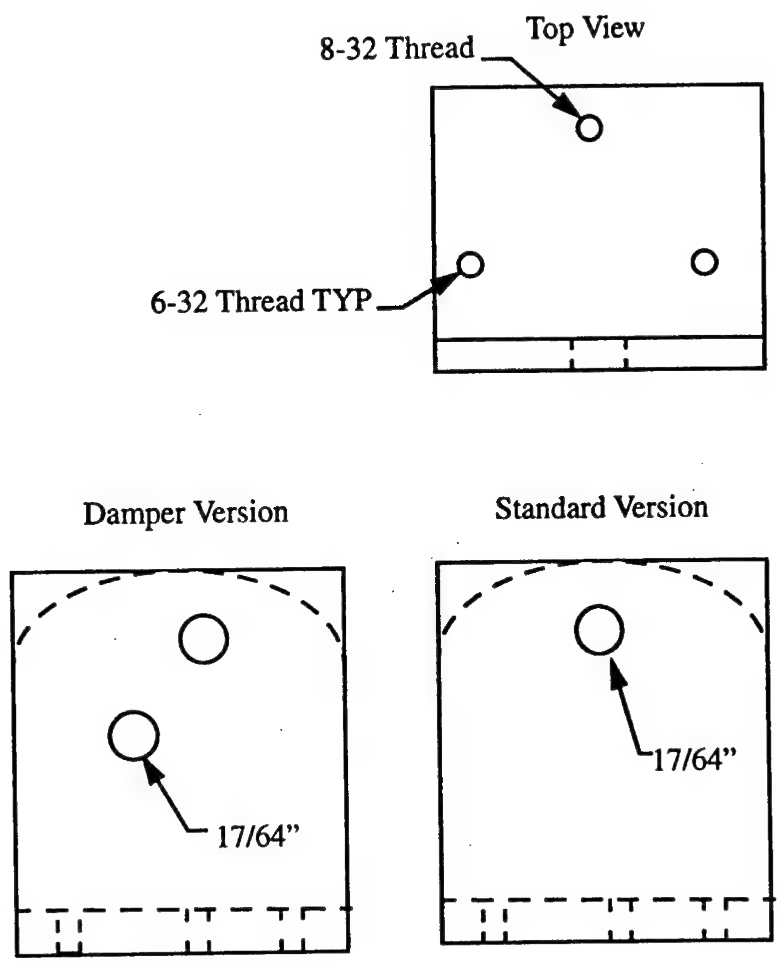


Figure D.1: Spring/Damper Mount

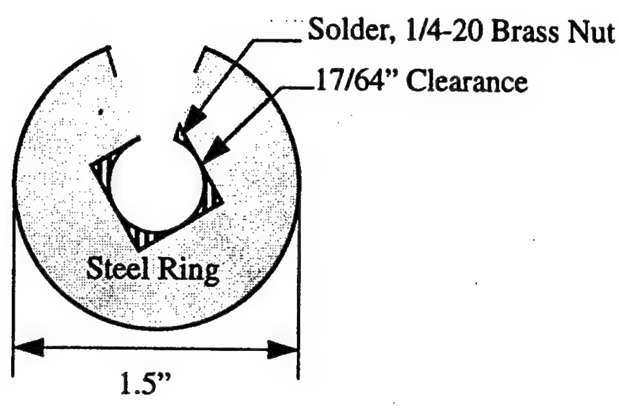


Figure D.2: Spring Holder

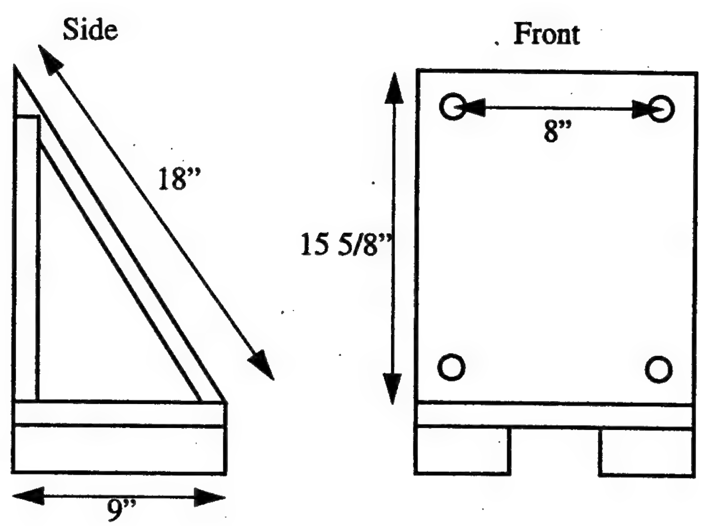


Figure D.3: Power Supply Stand

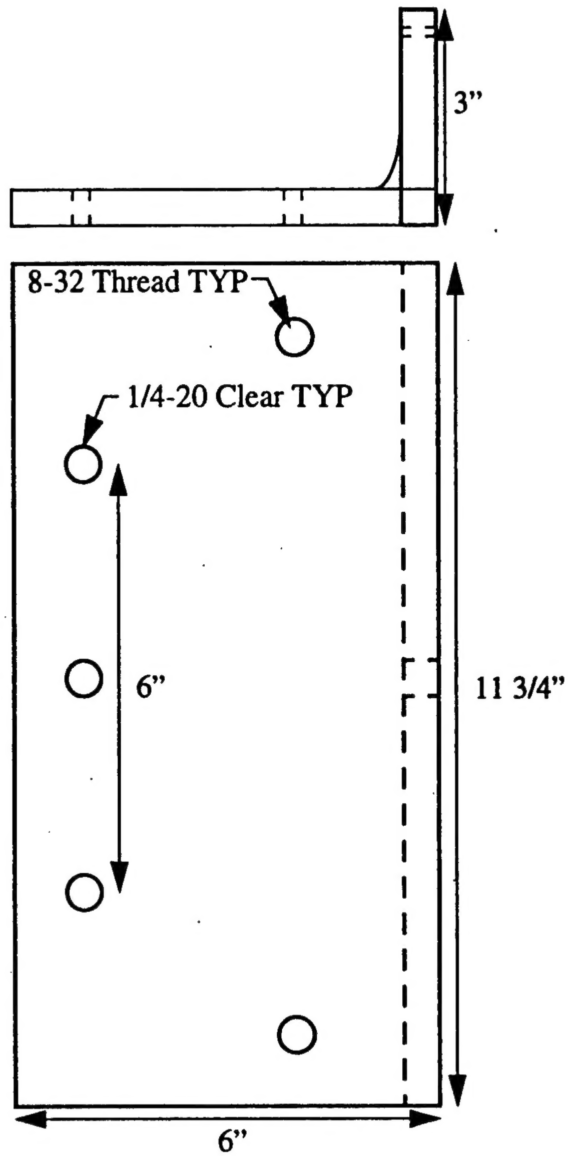


Figure D.4: Endpiece

University of Washington

Abstract

A Reconfigurable Hardware Testbed for Elastically-Coupled Systems

by Nicholas Scott Hardman

Chairperson of the Supervisory Committee: Professor Juris Vagners

Program Authorized to Offer Degree: Electrical Engineering

The product of this thesis is a hardware emulation testbed that can be used for hardware-in-the-loop simulations. In its most basic form, the testbed can be modeled as a rectilinear spring/mass/damper system. Rectilinear elastically-coupled systems are very traditional for academic interests. An ideal system is a series of aligned masses joined by linear elastic and/or inelastic bonds (e.g. springs and dampers) and placed on a frictionless surface.

Physical systems are valuable to both validate idealized models and explore the nonlinearities in 'real world' coupled systems. The same basic physical systems, but with feedback from position sensors, form closed loop systems that can test a host of controller architectures. In another scenario a physical system of the type already mentioned can be modified by using position sensors and computer interface software to drive a reaction mass actuator (RMA). The RMA is mounted midsystem and coupled to the system on both sides. In this way, the system becomes the physical representation of a general n^{th}-order system confined to rectilinear displacement. In

this form the motor becomes the source of both a constant input and deterministic (or stochastic) disturbance. The RMA is used as the control actuator to dampen disturbances while passing the desired input signals. In this scenario the inputs can be multiple frequency signals convolved in the computer and fed into a single actuator.

The three configurations mentioned fall into three areas of control study: system identification, regulation/tracking, and input disturbance rejection. All three areas of study can be performed on RECS, the hardware testbed at the University of Washington Control and Robotic Systems Lab. The testbed was designed with a large amount of variability in all parameters. It is also capable of variable-order configurations, so it has the capability of modeling a wide range of systems. The closed loop forms of the testbed provide a dynamic simulation platform for feedback control and input disturbance studies.

REFERENCES

[1] Chrisman, Karl, "Coplanar Inverted Pendulum and Pivot Arm", Master's Thesis, University of Washington, 1994.

[2] De Silva, Clarence W., Control Sensors and Actuators, Prentice Hall, Inc., Englewood Cliffs, NJ, 1989.

[3] Franklin, Gene F., Powell, David J., Emami-Naeini, Abbas, Feedback Control of Dynamic Systems, 3rd ed., Addison-Wesley Publishing Co., New York, NY, 1994.

[4] Ling Dynamic Systems, "Installation and Operating Manual", 2nd ed., Amendment 9, Royston, England, Jan 1995.

[5] Saner, Floyd E., "Servo Motor Application Notes", Pittman, Harleysville, PA, 1990.

[6] Young, Hugh D., University Physics, 8th ed., Addison-Wesley Publishing Company, Inc., New York, NY, 1992.

[7] Sepp, Kalev, "Robust Control of Linear Track Cart-Pendulum", Master's Thesis, University of Washington, 1994.

[8] Leahy, Jennifer Martin, "Control of Nonlinear Underactuated Systems: The Rotary Arm Inverted Pendulum Problem", Master's Thesis, University of Washington, 1995.

[9] Pittman, "Bulletin 5000", Harleysville, PA, 1989.